the CP/M* and S-100 user's journal

$2.00
U.S.A.

MICROSYSTEMS™

NOV/DEC 1981

VOL. 2/NO. 6

INTRODUCTION TO THE C PROGRAMMING LANGUAGE

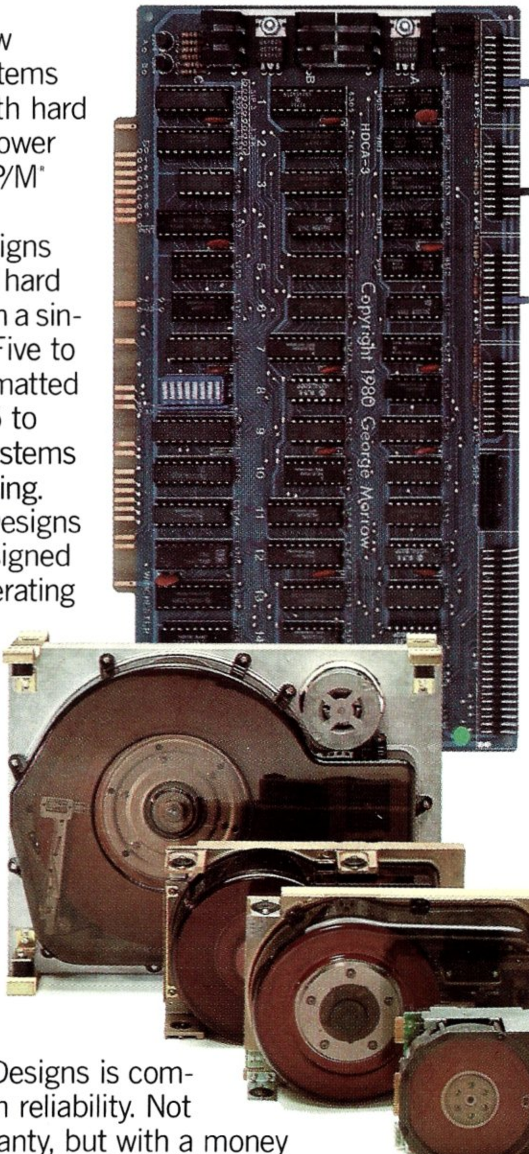
See Pages 20-38

Also in this Issue

MORROW DESIGNS

NOW, 26 Mbytes $4,495

Leading edge technology in hard disk systems.



Complete systems. Morrow Designs hard disk subsystems are delivered complete with hard disk, controller, cabinet, power supply, fan, cables and CP/M* 2.2 operating system.
Widest range. Morrow Designs offers the widest range of hard disk systems available from a single supplier. 5¼," 8," 14." Five to over 100 megabytes of formatted hard disk storage. $2,995 to $17,980. Cost effective systems that work. And keep working.
S-100 and more. Morrow Designs hard disk systems are designed for use with the CP/M operating system. Available software packages allow our systems to run on any IEEE696/S-100 Standard system with no hardware modification.
Plus, Cromemco,*** North Star,** Vector Graphics, Godbout, Dynabyte, Exidy,**** IMSAI, Micromation, Processor Technology and California Computer Systems.
Reliable systems. Morrow Designs is committed to hard disk system reliability. Not simply with a 90-day warranty, but with a money back guarantee. If our system fails to perform to specification, send it back. We'll send back your money.

Experience. As of April, 1981, there were over fifteen hundred Morrow Designs hard disk systems successfully installed. In fact, over 200 independent systems integrators now use our hard disks to solve their mass storage problems.
Performance answers. Morrow Designs hard disk systems have been benchmarked against all other systems. None is faster under CP/M. Morrow Designs hard disks operate at 10 times the speed of a floppy disk drive. Transfer rates range from 590,000 bytes to 900,000 bytes per second. That kind of performance can become addictive.
Cost effective answers. Compare Morrow prices and performance to anything presently available for S-100 systems. You'll find Morrow's price/megabyte/performance ratio to be unmatched. Leadership in disk systems technology earned us leadership in price/performance. And that may have earned us a call from you. Circle the Reader Service Number for our full line data sheets. Can't wait? Call us at (415) 524-2101. And yes, OEM quantity prices are available. **LOOK TO MORROW FOR ANSWERS.**

MORROW DESIGNS
5221 Central Avenue, Richmond, CA 94804
(415) 524-2101

*CP/M is a trademark of Digital Research.
**Northstar is a trademark of North Star Computers, Inc.
***Cromemco is a trademark of Cromemco, Inc.
****Exidy is a trademark of Exidy Corporation.

features:

- Virtual Memory Management
- Motorola 68000 Processor
- 12 Mhz operation
- 32 bit computations
- 4 Billion byte addressing
- LRU demand paging
- Peripheral processor
- UNIX multi-user operating system

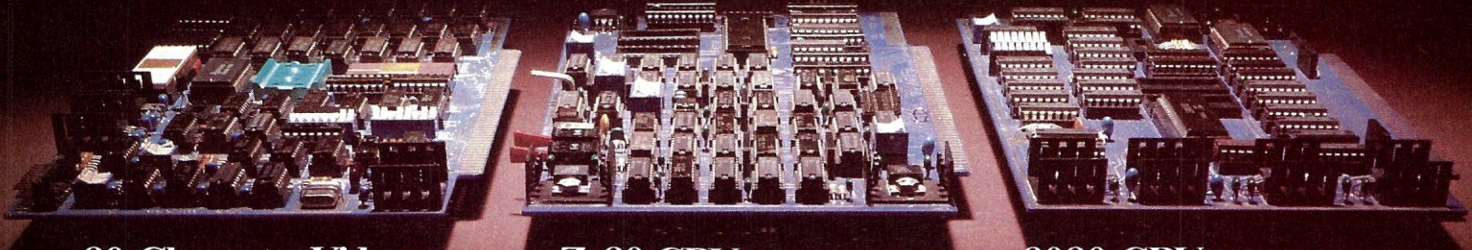
Only the world's finest minicomputer offers these features in systems available TODAY. Complete single-user systems start under $10,000. And a 6 user UNIX-based 16 MB virtual system with 512K Bytes RAM and 24M Bytes hard disc costs under $25,000! Want proof? Call or write now for brochures and information on our 68K Seminars. Or send $75 for the complete technical manual set.

MicroDaSys
2811 Wilshire Blvd., Santa Monica, CA 90403
Call (213) 829-6781

TWX: 910-321-2378

A free poster of this ad is available while supply lasts.
UNIX is a trademark of Bell Laboratories.
68K MiniFrame is a trademark of MicroDaSys, Inc.

A Busload from SSM.



80 Character Video

With 80 characters per line our VB3 is the perfect video interface for word processing. It produces a standard 80x24 display of upper and lower case characters or as much as 80x50 for a full page of text. The matrix for graphic display goes up to 160x200. And with optional EPROM, as many as 256 user programmed characters or symbols can be produced.

VB3 is memory mapped for rapid screen updating. But it occupies memory only when activated. So one or more VB3s can be located at the same address with a full 65K of memory still available to the user.

It generates both U.S. and European TV rates and meets IEEE 696.1 standard. Other features include keyboard input, black on white or white on black, one level of grey, underline, strike thru, blinking char., blank-out char., and programmable cursor. Software includes a CP/M compatible driver and a powerful terminal simulator.

Z-80 CPU

We spent over a year designing the CB2 to assure that it is the most fully S-100 compatible Z-80 CPU on the market.

It operates at 2MHz or 4MHz by DIP switch selection and includes two sockets for 2716/2732 EPROMs or 2K RAMs. Memory sockets can be disabled. Separate run/stop and single step switches allow system evaluation without the benefit of a front panel.

CB2 also features an MWRITE signal, firmware vector jump, and an output port to control 8 extended address lines (allowing use of more than 65K of memory). Jumper options generate the new IEEE 696.1 signals.

8080 CPU

Our CB1A is identical to our popular CB1 with the exception that the on-board RAM has been increased from 256 bytes to a full 1K.

It also features an optional 2K of 2708 EPROMs, power-on/reset vector jump. MWRITE, parallel input port with status and DIP switch addressing.

Our S-100 line includes CPU, Video, I/O, 8 and 16 Bit RAM, EPROM, EPROM Programmer, Prototyping, Terminator, Extender, and Mother boards. Available assembled or as kits.

New SSM Products.

Please call for all the latest details on our newest products:

- **MB 64** 64Kb static RAM memory.
- **I05** two parallel, two serial input/output ports, with complete RS-232 protocol.
- **I08** multiple RS-232 serial I/Os.

SSM

SSM Microcomputer Products, Inc.

2190 Paragon Drive, San Jose, CA 95131, (408) 946-7400 Telex: 171171, TWX: 910-338-2077

the CP/M* and S-100 user's journal

MICROSYSTEMS™

Volume 2, Number 6 November/December 1981

Staff

Sol Libes	editor
Claudette Moore	managing editor
Chris Terry Jon Bondy Anthony Skjellum	contributing editors
Don Libes Fred Gohlke David Gewirtz Randy Reitz	assisting editors
David Ahl	publisher
George Blank	editorial director
Laura MacKenzie	production manager
Sue Gendzwil	art director
Diana Negri	assistant art director
Chris Demilia Candace Figueroa Carol Ann Henderson Eugene Bicknell	layout
Jean Ann Vokoun Maureen Welsh	typesetting
William L. Baumann	operations manager
Patricia Kennelly	personnel and finance
Ethel Fisher	bookkeeping
Elizabeth Magin	secretary
Moira Fenton Frances Miskovich Terri Murphy Dorothy Staples Carol Vita	circulation
Ralph Loveys	customer service
Linda Blank Sue DeMark Gail Harris Mark Smith Jim Zecchin	order processing
Karen Brown	receptionist
Jennifer Burr Laura Gibbons	retail sales
Mary McNeice	office assistant
Ron Antonaccio	shipping & receiving

Editorial correspondence is welcomed and should be sent to: Sol Libes, c/o MICROSYSTEMS, Box 1192, Mountainside, NJ 07092. Phone: (201) 522-9347.

For information on commercial advertising, write to: MICROSYSTEMS, 39 East Hanover Ave., Morris Plains, NJ 07950, or call Claudette Moore at (201) 267-4558.

In This Issue

MICROSYSTEMS (ISSN #0199-7955) is published bi-monthly by Creative Computing, 39 East Hanover Ave., Morris Plains, NJ 07950. Second class postage paid at Morris Plains, NJ 07950 and additional mailing offices.

Subscriptions are $10 a year (6 issues USA). To subscribe, call our toll free number: (800) 631-8112, or (201) 540-0445 in New Jersey.

POSTMASTER: Send address changes to: MICROSYSTEMS, P.O. Box 789-M, Morristown, NJ 07960.

NEWS & VIEWS

by Sol Libes

Digital Research Releases MP/M-II

Digital Research has introduced MP/M-II, which is specifically designed for multi-user systems. MP/M-II features file and record locking, password protection and increased disk capacity. The first version of MP/M, which was released in 1979, was well suited for multitasking applications but lacked provision for preventing concurrent access of a data record by more than one user.

Altos Computer Systems was the first to sign a license agreement for MP/M-II for its four-user system. Altos plans to enhance their version with interrupt-driven console I/O and other features to improve performance. Digital Research claims that as many as fifteen software firms have declared their intentions to extend their software to run under MP/M-II.

MP/M-II will run on an 8-bit system, with 48K RAM and clocked interrupt. It can support up to sixteen users, sixteen printers, sixteen disk drives (512Mbytes each max., 8Gbytes max.) and 400Kbytes of RAM. RAM can be divided among eight users in 48K banks. The MP/M-II package includes RMAC (relocatable macroassembler), LINK-80 (linker) and LIB (program library management utility). It is compatible with CP/M files (up to 32Mbytes max.) and supports CP/NET, allowing interconnection of multiple systems. MP/M-II is priced at $450 and OEM discounts are available.

Digital Research Publishes Software Catalog

Digital Research has published a *CP/M Compatible Software Catalog* which lists 116 independent software vendors (ISV) who write CP/M compatible software. The 24 page catalog is $5, and is available from: Digital Research, Box 519, Pacific Grove, CA 93950. ISV's who would like to be listed in the catalog should write to Digital Research asking for the appropriate forms to be sent.

Digital Research is rumored to be working on software based on the Motorola 68000. Most likely the first package to be released will be MP/M—look for it to be released in the last quarter of next year. Further, you can expect enhancements to MP/M-II to be released soon. Also due soon from the language group (formerly Compiler Systems) will be a true Basic Compiler producing object code that runs without a run-time program. An 8080 version is due in early 1982 with an 8086 verson to follow in mid- or late 1982. The Basic Compiler will have several enhancements, including labels for writing structured-type programs. The group also plans other languages in addition to the present Basic and PL/I compilers. It is also rumored that CP/M 3.0, originally expected by year-end, has been put on a back burner.

North Star Unwraps Advantage & New Horizons

North Star has introduced several new computers: the "Advantage," a $4,000 desktop computer with graphics ability and two new multi-user systems based on the Horizon mainframe. The Advantage uses a Z80A with 64K-RAM plus 20K-RAM graphics memory (240 x640 pixels). It also features two double-sided, double-density mini drives, an 87-key keyboard (fifteen programmable keys) and a 12" screen. The Horizon systems support up to five users, and use either a 5Mbyte or 18Mbyte winchesters. Local networking interfaces are planned for all three systems.

Typesetting From CP/M ASCII Files

In one of my recent editorials I had mentioned sending a CP/M text file over the phone line to a typesetter, and having type set directly. Several readers wrote to me asking for the name and address of such a typesetter. I am sure that you can locate one in most of the larger metropolitan areas (just look in the yellow pages). If you cannot find one in your area, then I recommend you try Dale Brown, Brown Graphic Press, 2488 Summit St., Columbus, Ohio, 43202.

You will have to add typesetting control codes to the text that specify type size, spacing, style, etc. You must pay an initial fee of $50 to cover the cost of a manual and the proofing and correction of your first few submittals. Turnaround time is about two working days (including proofing), and about one day when no proofing is required. They can accept text on TRS-80 Scripsit or CP/M 8" disks. The modem service operates 24 hrs/day and you can save money by sending the file after 11PM.

Marc Author Killed in Accident

Ed Ziemba, author of the Marc Operating System (see the Jul/Aug 1981 *Microsystems*, page 12) died in June in a snorkeling accident. Despite this tragic loss, Leor Zolman, who was working with Ed on Marc, said that he and his associates plan to do the final polishes of the software and will release it shortly.

JANUS

the language that is based on the past but looks to the uses of the future.

ADA

Now You Can Have **JANUS** On Your CP/M System.

JANUS is a subset of the famous **ADA** language developed by the U. S. Department of Defense. This new language was designed to ease maintenance of programs, yet be easy to program in. **ADA** combines the best features of **ALGOL** and **Pascal** to produce a powerful programming language. **ADA** puts an emphasis on Modular programming and program portability.

The **JANUS** implementation of an **ADA** subset includes features found in no other micro-processor programming language. Features like true modular programming, where the computer, rather than the programmer, remembers which modules are being used. Features like re-entrant initialized variables. Features like full error walk-backs and line number error messages. And much, much more.

The **JANUS** system comes complete — no other programs to buy — *It includes:*

- Compiler
- Assembler
- Disassembler
- Screen Editor
- Linker
- Full Documentation

The JANUS package produces ROMable, Re-entrant 8080 or **optimized Z80** machine code.

You too can take a step forward into the future.

ORDER before December 31 and get in on the special introductory price of $250.

JANUS requires an 8080/8085 or Z80 CPU, 56K of memory, and at least one 8″ floppy disk. **JANUS** is supplied on a single density 8″ floppy disk. Specify 8080 or **Z80 optimized** version.

RR SOFTWARE

specialists in state of the art programming

P.O. BOX 1512 MADISON, WISCONSIN 53701 (608) 244-6436

Cambridge Development Lab Offers Newsletter And Seminar

Cambridge Development Laboratory is offering free subsrcriptions to its "Dynamic Blackboard News," a newsletter covering applications, hints and technical notes for graphics users. CDL will also present a one-day seminar titled "Introduction To Microcomputer Graphics" in Boston on November 14th ($20/person). To obtain more information on the seminar, or to receive the newsletter contact: Jean L. Graef, CDL, 36 Pleasant St., Watertown, MA 02172; (617)926-0869.

ISIS ⟷ CP/M®

Full bi-directional file transfer capabilities are provided in the ISIS-CP/M utilities package. Written in machine language and running under CP/M, these utilities permit the CP/M user to read or write files direct to/from an ISIS Diskette. They will run under any version of CP/M without regard to diskette density. The complete package is $250.00 including user's manual. Write for free brochure on other CP/M software.

CP/M is a registered trademark of Digital Research
ISIS is a trademark of Intel Corporation

SOUTHERN COMPUTER SYSTEMS, INC.
P.O. Box 3373A
Birmingham, AL 35255
(205) 933-1659

North Star
SMALL BUSINESS SOFTWARE

ACCOUNTING – 3 programs that record, compare and total business expenses

INVENTORY – 2 programs for inventory control

GENERAL – 5 general purpose business programs

CONSTRUCTION – 1 program for building construction estimates and material ordering

send for free catalog explaining these programs

CI COMPUTING INTERFACE
1918 Carnegie Lane #c
Redondo Beach, Ca 90278
(213) 372-0540

User Groups News

The SIG/M group (Box 97, Iselin, NJ 08830) has released seventeen new volumes, bringing their total to 42 volumes. Thirteen more new volumes are in development, and are expected to be released by year-end. The CPMUG group (1651 Third Avenue, New York, NY 10028) has released four new volumes, bringing their total to 54 volumes. A full report on these two user groups will be published in the next issue of MICROSYSTEMS.

The Connecticut CP/M User Group meets on the last Monday of each month at 7:30PM at the Wordsmith Network, 110 Day Hill Rd., Windsor, CT 06095. For more information call Charlie Lowderback at (203)683-2427. An attendance fee of $2/meeting is charged.

"TCS Debits & Credits" is a new newsletter for users of TCS accounting systems software. It contains new product and version announcements, bug reports, corrections, and enhancements on TCS products. It will be published quarterly, and a subscription is $20/yr. For more information contact: TCS System Users Group, c/o Mountain Software Systems, Box 3282, Walnut Creek, CA 94598, (415)625-1592.

The Amethyst User Group has been formed to support the Mince, Scribble, and Amethyst text editor and formatter. The main purpose of the group is to provide coordination among users developing extensions to Mince and Scribble. Membership is $6/yr. For more information contact Barry A. Dobyns, 1633 Royal Crest #1128, Austin, TX 78741, (512)441-9466.

The Osborne Business Software Users Group has been formed at: 2256 Main St., Suite 11, Otay, CA 92011; (714)423-0538. Membership is $10/yr domestic, $20/yr foreign.

I recently received issue #8 of the Pascal/Z User Group Newsletter. The issue was 31 pages long, and it keeps improving. The group supports InterSystems Pascal/Z and has also just released Volume #11 of their public domain software. For information write: Pascal/Z UG, 7962 Center Parkway, Sacramento, CA 95823.

An Intertec Users Group has been formed called "Super*Star." It will publish a monthly newsletter and furnish other services. For information, contact: Amy Roosevelt, 3722 Chestnut Pl., Denver, CO 80216, (303)623-7973.

Digital Research has formed a PL/I-80 User Group (PLUG). PLUG members will receive the PLUG and ISV newsletters (Independent Software Vendor). A PLUG library has also been formed, consisting of source code programs submitted by PL/I-80 users. Two disks are already available to members at $25 each. Users who submit acceptable programs to the library or attend an ISV seminar will receive a year's free membership. For more information write: PLUG, Digital Research, Box 579, Pacific Grove, CA 93950.

The BDS-C Users Group (Box 287, Yates Center, Kansas 66783) has released three new volumes of software. One volume contains an updated version of Small-C (in source) which includes an assembler, linker, submit and clock board driver programs. The other two volumes contain utilities and a modem program.

New Bits

A 10-page quick reference guide to Oasis Basic is available for $1.00 from Phase One Systems Inc., 7700 Edgewater Dr., Suite 830, Oakland, CA 94621...........The second edition of the "CP/M Software Index" has been published listing 740 programs offered by 248 software vendors ($6 North America, foreign $8, CA residents add 6% tax). Order from: Small Systems Group, Box 5429, Santa Monica, CA 90405, (213)392-1234......Lifeboat Associates, the world's largest publisher/distributor of CP/M-based software has announced that it will soon begin distributing software for the new IBM personal computer DOS developed by Microsoft.

Forth Interest Group National Convention

The FORTH Interest Group (FIG) will hold a National Convention November 28, 1981, 9am - 6pm, at the Santa Clara, California Marriot Hotel. The one day convention will include a presentation, workshops, hands-on equipment and a number of vendor exhibits. An evening dinner will follow the day's activities and will feature a speaker. Registration is $3.00, dinner is extra.

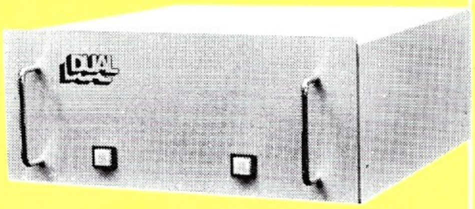
For information: FORTH Interest Group, P.O. Box 1105, San Carlos, CA 94070, (415)962-8653. ■

POWER & RELIABILITY

68000 μP on the S-100 Bus?

YES, AVAILABLE NOW FROM DUAL SYSTEMS!

- ☐ **8 mHz 68000 microprocessor.**
- ☐ **16-megabyte direct addressing.**
- ☐ **32-bit internal arithmetic.**
- ☐ **Minicomputer type instructions** including MULTIPLY.
- ☐ **FULL IEEE-696 S-100 compliance.** Runs with all 4 mHz S-100 boards and automatically runs faster when accessing Dual Systems memory boards, for FULL SPEED OPERATION OF THE 68000.
- ☐ **Powerful vectored interrupts.** 7 Vectored interrupts, including NMI, as well as alternate mode having up to 256 interrupts.
- ☐ **On board monitor ROM** for immediate use.
- ☐ **Connector for future addition of memory management** unit for multi-user operating systems.
- ☐ **Built to the highest industrial standards** with 200 hour burn-in.



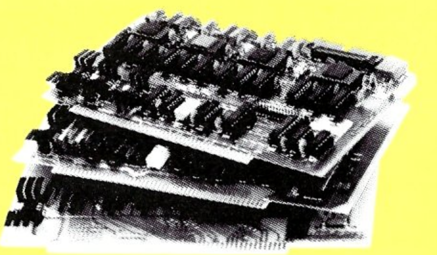
CPU/68000 CPU board..... $1195

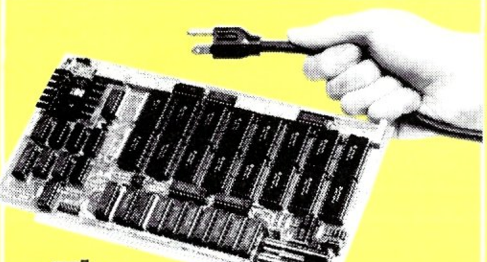
32K-byte 8/16-bit NONVOLATILE RAM board, for secure storage of programs you are developing. Allows FULL SPEED CPU operation. CMEM-32K, per 32K-bytes .. $895

32K-byte 8/16-bit EPROM board, EPROM-32K **$245**

Serial I/O board, SIO-2 ...... $285

All of the above with cabinet, power supply and backplane..... $3685





The Ultimate IEEE S-100 Memory Would...

- ☐ **BE NONVOLATILE** holding data for up to eight years with the power off.
- ☐ **RUN AT 6 MHZ** without wait states.
- ☐ **RUN IN 8 OR 16-BIT** systems with 8 or 16-bit wide data paths.
- ☐ **HAVE EXTENDED 24-BIT ADDRESSING** and bank select.
- ☐ **HAVE DYNAMICALLY MOVABLE WRITE PROTECT AREAS** to prevent accidental erasure of programs and critical data.
- ☐ **GENERATE POWER-FAIL** interrupts for orderly system shutdown & power failure recovery.

...Available Now from Dual Systems

The Dual Systems CMEM memory boards combine high-speed CMOS memories with new 5-10 year lithium batteries to give you the nonvolatility of an EPROM board while retaining the instant writability of a high-speed read/write RAM. These industrial grade boards are ruggedly built and are burned-in for 200 hours.

☐ CMEM-32K, 32K-bytes ... $895

☐ CMEM-16K, 16K-bytes.... $795

☐ CMEM-8K, 8K-bytes...... $695

Sales representatives in most metropolitan areas.

OEM and Dealer pricing is available.

Toughest Boards in Town... IEEE 696/S-100

SUPER RELIABLE NON-STOP CLOCK

Keeps time with power off. Our industrial clock utilizes a new lithium battery for 3-9 years use. Easiest clock to program you'll ever see. Runs in all S-100 systems.

☐ Year, date, hrs, mins, secs, msecs.
☐ Uses new LSI CMOS chip.
☐ Vectored interrupts.
☐ CLK-24 .... $250

A/D CONVERTER

IEEE696/S-100 AIM-12 industrial standard module designed for industrial analog-to-digital use.

☐ Runs in all S-100 systems.
☐ 32-channel, 16-differential ☐ 12-bit resolution/accuracy. ☐ 25-microsecond conversions.
☐ Instrumentation amplifier.
☐ BASIC program provided. ☐ AIM-12, $695 or $785 w/1-1000 gain transducer amplifier.

D/A CONVERTER

AOM-12 IEEE696/S-100 industrial level digital-to-analog (D/A) converter.

☐ 12-bit ± 1/2 L.S.B. accuracy over full 0-70°C temperature range.
☐ Outputs 0-10, ± 5, or ± 10 volts.
☐ Short circuit protection, all outputs.
☐ Switch-programmable for multiple boards.
☐ AOM-12, $575

VIC 4-20

Standard output for industrial control 4-20 mA D/A converter. Used in conjunction with the D/A board.

VIC4-20, $445.

DUAL 77 Data Acquisition and Control System-

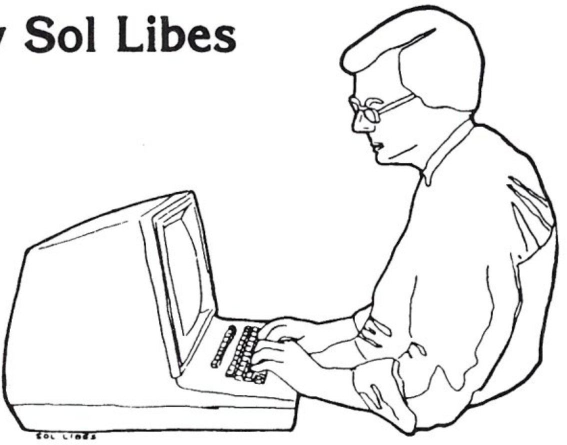
Built to industrial standards; designed for severe environments. BASIC language makes programming easy. Access to hundreds of sensors. Expandability to meet your increased needs. Nonvolatile memory. Power interruption recovery with automatic restart. DUAL 77 is economical; $5985 & up.

DUAL

DUAL SYSTEMS CONTROL CORPORATION **system reliability/system integrity**

720 Channing Way • Berkeley • CA 94710 • (415) 549-3854 or (415) 549-3890 • TWX 910366 2035

by Sol Libes



EDITOR'S PAGE

Another Language Issue

This issue of *Microsystems* continues to emphasize languages. We had originally planned to have only one issue on this topic (Sept/Oct 1981) and to emphasize MP/M in this issue. However, we received so many articles on languages that we had to change our plans. In fact, we have received enough excellent articles to have a third issue on languages—but we will hold off on that one for awhile. We have therefore set the following tentative schedule for *Microsystems*. Although we have some articles on these topics already, we are searching for more good material. I would appreciate hearing from potential authors on these topics:

Jan/Feb 1982	PL/I-80
Mar/Apr 1982	Multi-User Systems
May/Jun 1982	Data Base Systems
Jul/Aug 1982	68000 & Z8000 Systems
Sep/Oct 1982	Languages III
Nov/Dec 1982	Local Networks

IEEE-696/S-100 Bus Standard Status Report

I regret to report that the final revision to the IEEE-696/S-100 Bus Standard was not entirely ready as we were putting this issue together, early in September. We expect to print the entire standard as soon as it becomes available.

To print Or Not To Print, That Is The Question

As editor of *Microsystems*, I have on a few occasions rejected an article that I really would have liked to publish. The problem was that the article included source code that would have taken up an entire issue, and sometimes more than an entire issue. It was with deep regret that I rejected these articles.

Although the magazine has nearly doubled in size from the first issue published less than two years ago, editorial space in the magazine is still at a premium. We have only a certain number of pages allocated in each issue for articles.

Therefore, I am seriously considering omitting source code listings from the magazine when such listings are more than two or three pages long. Instead, we would refer the reader to the author to obtain the source code on disk. After all, who really wants to key in a lot of code with its attendant entry problems? Personally, I certainly would be willing to pay a reasonable sum to save the time and entry problems. Also, it would free up editorial space in the magazine so that we could include more articles.

Before making such a radical change, I would like to hear from Microsystems readers and authors. Are you in favor, or opposed to this change? What do you feel is a reasonable charge for the source code? Please let me hear from you at: Box 1192, Mountainside, NJ 07092.

IEEE-696/S-100 Interfacing Book Published

Mark Garetz and I have written a new book entitled *Interfacing To S-100/IEEE-696 Microcomputers* which has been published by Osborne/McGraw-Hill ($15.00). Naturally, it is impossible for me to discuss this book without prejudice since I am one of the authors. I feel however, that it is authoritative and comprehensive. It presents in-depth discussions of the S-100 bus signals, timing relationships, memory and I/O interfacing, interfacing to "real world" devices, digital-to-analog and analog-to-digital conversion, S-100 interrupt handling, using programmable timer/counters and setting up multi-master systems under the IEEE-696 TMA protocol. All circuit diagrams are complete with part numbers, values and pin numbers so that the circuits can be wired directly from the book. Several dozen software driver routines are also presented in the 321 page book for use with the circuits.

Digital Research Makes Major Changes

Digital Research is undergoing major changes in organization. The company is formalizing its management structure, accepting venture capital financing, altering its pricing policies and moving to a new location.

Gary Kildall created CP/M in 1974 when working as a software consultant to Intel. Intel decided not to use it, so Gary formed Digital Research Inc. in 1975 to market CP/M. IMS took a look at it and decided to use it as the operating system for their Imsai-8080 microcomputer. It was an immediate success.

THINK DIGITAL MARKETING. THINK AHEAD.

NOVEMBER

2670 Cherry Lane • Walnut Creek, CA 94596 **(415) 938-2880**

MILESTONE™—$295. Manual alone—$30. "Critical path" network analysis program for scheduling manpower, dollars and time to maximize productivity.
NEW IMPROVED. Interactive project management program that runs under CP/M. MILESTONE can be used to track paper flow, build a computer, check a department's performance, or build a bridge. MILESTONE can be used by executives, engineers, managers, and small businessmen.
- Produce PERT chart in minutes.
- Find critical tasks that can't be delayed.
- Investigate tradeoffs between manpower, dollars and time.
- Give plans to others using a printed project schedule.
- Change details and immediately see the results on screen.
- Balance time, manpower and costs.

Requires 56K RAM and CP/M. Specify Z80 or 8080. Also available for Apple Pascal, UCSD Pascal or CP/M-86 operating systems. Formats: 8, NS, MP, SB, TRS2, APPL, OB-1, XX, I-5

ACCESS/80™
A report generator and cross-tabulator. Virtually any report that can be described on paper can be generated by using your existing ASCII data files. Produces reports in minutes that would take hours to program in BASIC.

Level I—Report Generator and Cross-tabulator—$295.
Read ASCII files and create sorted reports with subtotaling capability. Provides multi-dimensional cross tabulation and computation. Includes operating system commands.

Level II—Output and Logic Processor—$495.
Everything in Level I plus, write out new files in any sorted order (including subtotaling). Load arrays from files. Performs binary search on sorted arrays in memory. Includes control language extensions for complex applications.

Level III—Information Management System—$795.
Available 1st Qtr '82
Everything in Level I and II plus full screen data entry and updating, and Btree indexing.
Requires CP/M and 48K RAM. Formats: 8, NS, MP, CDOS, SB, TRS2, APPL

DATEBOOK II™—$295. Manual alone $25.
- Schedules appointments for up to 27 different doctors, lawyers, rooms, etc.
- File structure allows for appointments up to one year in advance.
- Searches for openings that fit time of day, day of week and/or day of year constraints.
- Appointments made, modified or cancelled easily.
- Copies of day's appointments can be printed quickly.

Requires 56K RAM and CP/M. Specify Z80 or 8080. Also available for Apple Pascal, UCSD Pascal or CP/M-86 operating systems. Formats: 8, NS, MP, SB, APPL, TRS2, OB-1, XX, I-5

PLAN80™—$295. Manual alone—$30.
A financial modeling system that's easy to use and powerful enough to replace your timesharing applications. Lets you calculate IRR and depreciation as well as trig functions effortlessly. You write a PLAN80 model just the way you would write a letter using any editor or word processing program.

PLAN80 results can be incorporated into any report that requires a financial model. It also tackles any numeric problem that can be defined on a worksheet. You'll remember **how** you created the model because calculations are defined using real English—not matrix coordinates.

At any point in the PLAN80 model you may display or print the results on your screen, printer or disk. Save all or part of the results for use by other models, or play "what if?" by inputting new values interactively.
Requires 56K RAM and CP/M. Also available for CP/M-86. Specify Z80, 8080, or 8086. Formats: 8, NS, MP, SB, OB-1, XX, I-5.

CBASIC/86™—$325. Manual alone—$30.
Industry standard intermediate code basic compiler with run-time interpreter for CP/M-86. Features include chaining, integer and external precision arithmetic, random and sequential records of any length (not limited to 256 bytes).
Requires CP/M-86. Formats: 8, I-5.

PROPERTY MANAGEMENT PROGRAM—$895.
Tracks income properties from house rentals to apartment complexes, shopping centers or office buildings. Permits entry of all expenses in any of 99 different categories for over 150 separate properties.

Now you can generate operating statements in minutes including totals of receipts and expenditures by category. Need more reports? This package includes a summary of transactions report, vacancy report, late rent report, rent roll report, check register and property management fees report.
Requires CP/M, 48K RAM and CBASIC2. Formats: 8, NS, SB, TRS2, APPL, MP

Personal PEARL™—$295.
User friendly application software generator. A fourth generation language, automatically creates completely new programs in CBASIC. Excellent documentation makes it easy to create "personal" software such as checking and financial programs, name, address and telephone lists, etc.
Requires CP/M, CBASIC2 and 48K RAM. Formats: 8, NS, SB, TRS2, APPL, OB-1, MP

MICROSTAT™ VERS. 2.0—$295. Manual alone—$25. Now includes stepwise and more flexible file structure. Code size smaller (=more data area). Also a data management subsystem for editing, sorting, ranking, lagging, data file transfers PLUS eleven data transformations (e.g. linear, reciprocal, exponential, etc.)
- Frequency and 8 probability distributions
- Multiple regression, correction analysis
- Time series
- Crosstabs/Chi-Square, non-parametrics
- Factorials, permutations, combinations
- Scatterplots
- ANOVA (one and two-way)

Requires 48K RAM, Northstar Basic or CP/M & CBASIC2 or Microsoft Basic 80. Formats: 8, NS, MP, SB TRS2, CDOS, APPL, XX

SPELLBINDER™—$495. Manual alone—$50.
Full feature word processing system with Office Management capabilities. Its special features include ease-of-use by office personnel, flexible print formatting & output, and a powerful macro capability which allows features to be added for the unique requirements of each user. Mail list macro is included for mail merge with form letters. Requires CP/M & 32K RAM. Formats: 8, NS, MP, CDOS, SB, APPL, XX

ACT I™—$125. Manual alone—$15.
CP/M compatible macro assembler for Z80, 8080/85, 6502 & 6800.
Assembler family supporting all major 8 bit micros. ACT features include full macro capabilities, comprehensive pseudo-ops, link-file structures, cross reference map, and algebraic expression processor. Requires 24K RAM & CP/M.

ACT II—$175. Manual alone—$20.
CP/M 2.x compatible cross assembler for 8086/88

ACT III—$125. Manual alone —$20.
CP/M 2.x compatible cross assembler for 6809.

ACT I and ACT II together—$225.
Formats: 8, NS, CDOS, MP/M, TRS2, APPL, OB-1, XX

PASCAL/M™—$225. Manual alone—$20.
CP/M compatible language for 8080/Z80 CPUs, supports full Jensen & Wirth plus 45 extensions to Standard Pascal including Random access files, 40 segment procedures & 16 bit BCD real type. Also includes symbolic debugger which features trapping on stores, examining and changing variables and tracing of program execution.
Requires CP/M 2.2 & 56K RAM. Formats: 8, NS, APPL, TRS2

PASCAL/M™ for 8086/88—$270. Manual alone $20.
All the features of Pascal/M for the 8086 and 8088 processors running under CP/M-86.
Requires CP/M-86 and 128 K RAM. Formats: 8, I-5.

SUPERCALC™—$295.
Allows a layman to manipulate business data in a variety of forecasting and accounting applications. Combines the interactive nature of an electronic spreadsheet with the power and convenience of a simple simulation language. Video display can be scrolled over entire worksheet using cursor controls. Symbolic vector references eliminate repetitive low-level data manipulation commands. Easy to use menu driven "Help" commands. Requires CP/M and 48K RAM. Formats: 8, NS, MP, SB, APPL, TRS2
Call for terminal formats.

SUPERDOS™—$129.
Upgrade of CP/M 2.2 for Superbrain. Includes ADM/31, Hazeltine, or Superbrain Terminal emulation mode. Other new features include 132 character keyboard buffer, repeat on all keys, key click, user programmable numeric keypad, 30% disk read/write improvement, real time clock, baud rates to 19.2K on RS232 ports, printer handshake modes, 4 new utilities, and 4 fixes.
Requires Superbrain 3.0. Format: SB

TRANS86™—$125. Manual alone—$20.
8086/88 Translator for existing 8080/Z80 programs. New source code is easily edited and asembled using ACT II to produce hex code which is executed by 8086/88 CPU. Emphasizes the extensions and features available in the 8086/88.
Requires CP/M & 32K RAM. Formats: 8, NS, APPL, OB-1, XX

MORE SOFTWARE	PRODUCT/MANUAL ALONE
TEXTWRITER III	$125./$25.
SPELLGUARD	$295./$20.
MCALL	$ 85./n/a
PEARL LEVEL 3	$650./$50.
CBASIC2	$120./$25.
ULTRASORT-II	$175./$20.
FABS	$175./$20.
AMCALL	$ 95./n/a
SELECTOR/86	$650./$35.
SELECTOR IV	$550./$35
GLECTOR for SELECTOR IV	$450./$35.
S-BASIC	$295./$35.
baZic	$150./$25.
dBASE II	$695./$50.
MAGIC MENU	$ 75./n/a
w/Spellguard option	$ 95./n/a
SPELL MENU	$ 95./n/a
MONEY MAESTRO	$200./n/a

DEALER INQUIRIES INVITED

FORMAT CODES: 8 (8″ single density IBM soft-sectored) NS (NorthStar DD), MP (Micropolis Mod II/Vector MZ), SB (Superbrain 3.0), CDOS (8″ Cromemco CDOS), TRS2 (TRS-80 Mod II), APPL (Apple II), OB-1 (Osborne-1), XX (Xerox 820), I-5 (IBM 5¼″)

SOFTWARE
SOFTWARE
DIGITAL MARKETING
DIGITAL MARKETING

Access/80 trademark Friends Software, Money Maestro trademark Innosys, Pearl trademark Relational Systems, Pascal/M, ACT, Trans 86, SuperCalc trademarks Sorcim, CBASIC2, CBASIC/86 trademarks Compiler Systems Datebook II, Milestone, Textwriter III trademarks Organic Software Spellguard trademark ISA CP/M, CP/M-86 trademarks Digital Research Superbrain trademark Intertec Data Systems Microstat trademark Ecosoft S-Basic trademark Topaz Programming Spellbinder trademark Lexisoft MCALL, AMCALL trademarks Micro-Call Services Selector IV, Selector/86, GLector trademarks Micro-Ap Prices subject to change without notice for shipping add $7. per item California residents add sales tax Terms: Prepaid check Am. Exp. M/C, VISA or in US COD (UPS) Prices quoted do not include dealer installation & training Prices and availability subject to change without notice.
Telex 17-1852 (DIGMKTG WNCK) DEALER INQUIRIES INVITED. OUTSIDE CALIFORNIA CALL (501) 442-0864. INSIDE CALIFORNIA CALL (415) 938-2883.

more . . .

SPECTACULAR OFFERS

BASF **WABASH**

MAXELL **OPUS**

We stock the complete line of BASF diskettes, reel-to-reel tapes, mag cards, disk packs and cartridges. We also carry MAXELL, OPUS and WABASH products. All are 100% certified and fully guaranteed.

Box of 10 diskettes:	5¼″	8″
OPUS ss/sd	$20	$21
BASF ss/sd	23	24
WABASH ss/sd	23	24
MAXELL . TOO LOW TO QUOTE.	CALL	

5¼″-10 sector-now available

Sectoring must be specified.

5¼″ or 8″ Vinyl Storage Pages . . . . 10/$5

LIBRARY CASES

8″ Kas-sette/10. . . . . **$2.99**
5¼″ Mini Kas-sette/10 . . **$2.49**

HARDHOLE DISK PROTECTORS
Reinforcing rings of tough mylar protect disk hole edge from damage.

	5¼″	8″
Applicators	$3	$4
Hardhole Rings (50)	$6	$8

DISK DRIVE HEAD CLEANING KITS

Prevent head crashes and ensure error-free operation.
5¼″ or 8″ **$19.50**

SFD C-10 CASSETTES . . . . 10/$7
(All cassettes include box and labels.)

Get 8 cassettes, C-10 Sonic, and Cassette/8 Library-Album, as illustrated, for only **$8**

SNAP-IT POWER CENTER

Turns 1 outlet into 6. Wall mount or portable. Circuit breaker, lighted switch and UL approved.
4″x3″x2″ **$19.95**

We also offer printer ribbons, printwheels, type elements, equipment covers, power consoles, paper supplies, storage and filing equipment, furniture and many other accessories for word and data processing systems. Write for our free catalog.

VISA • MASTERCHARGE • MONEY ORDERS • CERTIFIED CHECK • FOR PERSONAL CHECKS ALLOW TWO WEEKS • C.O.D. REQUIRES A 10% DEPOSIT • CAL. RES. ADD 6% SALES TAX • MIN $2 SHIPPING & HANDLING • MINIMUM ORDER $10 • SATISFACTION GUARANTEED OR FULL REFUND

ABM PRODUCTS

8868 CLAIREMONT MESA BLVD.
SAN DIEGO, CALIFORNIA 92123

Toll Free 800-854-1555 Order Only
For Information or California Orders
(714) 268-3537

Editor's Page, continued...

In six years Digital Research has improved CP/M, added MP/M, PL/I-80, CP/M-86 and a number of software development packages. It has grown from a two person company (Gary and his wife Dorothy McEwan) to fifty employees. Sales for the fiscal year ending August 1981 were about $6 million.

Digital Research is now gearing up for what they expect to be a $10 million year and $100 million within next five years. First, they have given up a small share of ownership to four venture capital firms. The infusion of capital has allowed Digital Research to purchase Compiler Systems, developers of the CBasic compiler. Second, the money will make it possible for Digital Research to move out of the Kildall home and into a 16,000 square foot modern office building.

Digital Research has also changed the structure of its sales agreements. Past CP/M licenses required a one-time fee of $50,000 to microcomputer companies who wished to furnish CP/M. This proved to be a bargain for these vendors. Now Digital Research requires graduated payment based on the number of computers sold by the manufacturer. Digital Research claims to have licenses with about three hundred computer manufacturers. ■

ATTENTION S-100 USERS, OEMs & ISOs!

MM-103 IS THE ONLY MODEM FOR YOUR NEEDS!

In previous issues, we listed more than 50 reasons why PMMI MM-103 modems are superior, along with a list of satisfied users that is now too long to print. Quality, integrity and low cost have made the MM-103 America's most popular modem. PMMI was the first to gain FCC approval and meet IEEE-696 S-100 standards. You won't find another modem for the S-100 bus with a wider range of Baud rates, more extensive and controllable software and such an unbeatable warranty. And since PMMI has eliminated the need for an acoustic coupler or an RS 232 adapter, your connection is more reliable and you buy no unnecessary hardware.

SO DON'T DELAY!

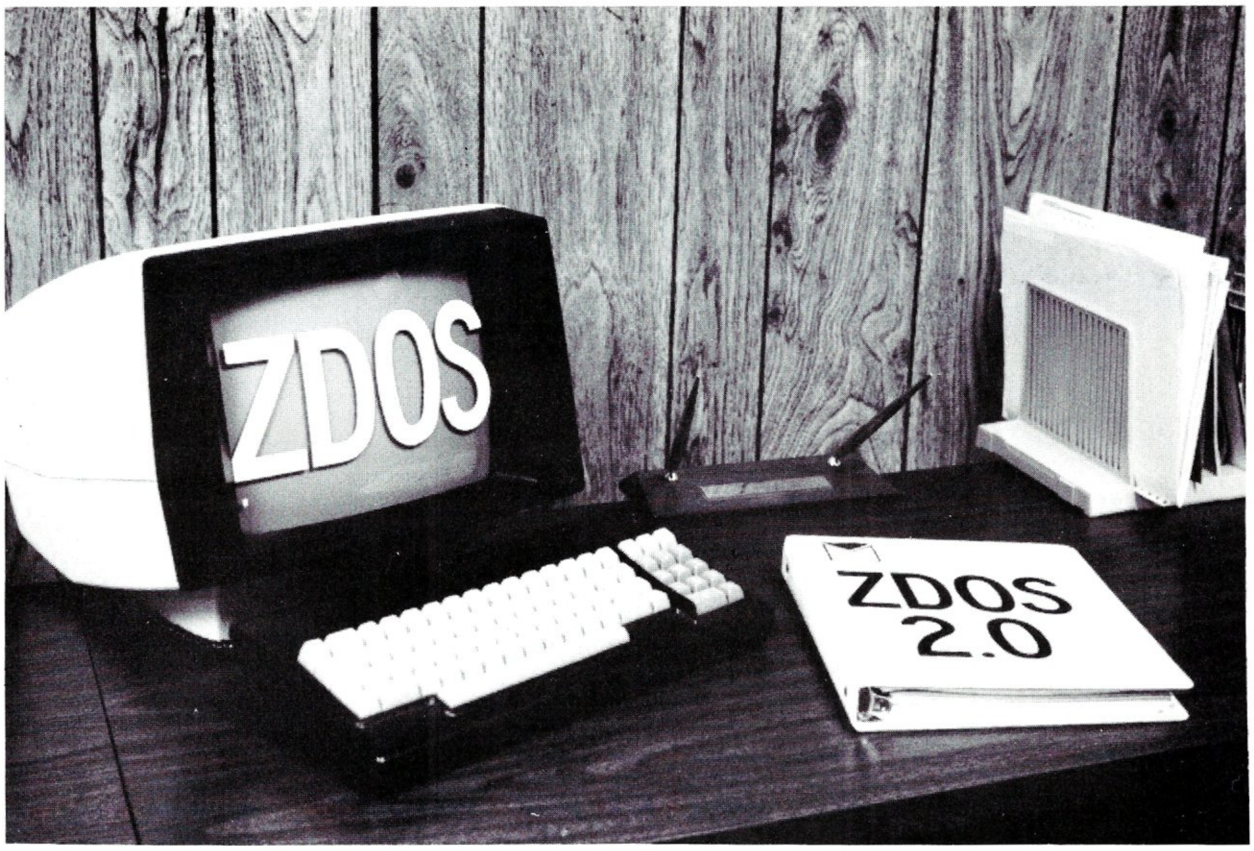
GO WITH THE MODEM WITH EXPERIENCE!

PMMI COMMUNICATIONS
[POTOMAC MICRO-MAGIC, INC.]

For further information, call or write:
Three Skyline Place
5201 Leesburg Pike, Suite 604
Falls Church, VA 22041
[703] 379-9660
Or dial into our 24 hour-a-day Modem Test Center:
[703] 379-0303 [300 Baud]

AFTER ALL... ALL MODEMS ARE NOT CREATED EQUAL!

STILL TIED TO A HOBBYIST ORIENTED O/S ?
UTILIZE THE FULL POTENTIAL OF YOUR Z-80 BASED SYSTEM
WHILE MAINTAINING COMPATIBILITY WITH CP/M *



NEW FROM MICRO-LOGIC
ZDOS™ 2.0

- More power than CP/M* without losing additional user memory.
- Full software compatibility with CP/M*.
- Supports floppy or hard disks.
- Up to 60% faster Disk I/O.
- Both maskable and non-maskable interrupt capability.
- Designed for networking or stand alone installations.
- Directory linking to other user numbers.
- Fully symbionic printer spooling.
- Unlimited nesting of SUBMIT files.
- Easy upgrade from CP/M* - use your present BIOS if desired.
- Clear, concise user reference manual.
- Full user support service for registered owners.
- Affordable Price.

MICRO-LOGIC

MICRO-LOGIC INC.
P.O. Box 4324
Hamden, CT 06514

ZDOS is a trademark of MICRO-LOGIC INC.
* CP/M is a trademark of Digital Research.

MICRO-LOGIC

MICRO-LOGIC INC
P.O. Box 4324
Hamden, CT 06514

☐ Please send further information on ZDOS.
☐ Send me ZDOS (manual incl.) Enclosed is my check for $149.95.
☐ Send ZDOS Manual only ($49.95).

NAME: __________

TITLE: __________

COMPANY: __________

ADDRESS: __________

____

PURCHASING STATUS: ☐ AUTHORIZE ☐ RECOMMEND
MY CURRENT SYSTEM 'CONFIGUATION:

HARDWARE: __________

____

OPERATING SYSTEM: __________

LETTERS TO THE EDITOR

Microsystems' Mix-up

Dear Editor:

A slight error crept into my article which appeared on page 42 of the March/April issue of *Microsystems*.

Those who made it to the third column (on page 43) must have been bewildered to find the sample code table (CODES1) where the code conversion instructions should have appeared. And, lo, where CODES1 belonged were the code conversion instructions. That section of the article should have read:

with the code character in the 'A' register:

```
D64A      SUI     FIRSTCH
FE0D      CPI     LIMIT + 1
D2nnnn    JNC     Error Routine
```

performs the conversion and validates the result. This leaves the offset in the 'A' register, so SELECTOR can be called as soon as the 'HL' register pair is set to the appropriate table.

If a code sequence isn't practical, the assembler will create a code table:

```
CODES1  DB   'E'   Code to be converted to zero
        DB   'P'   Code to be converted to 1
        DB   'T'   Code to be converted to 2
        DB   'D'   Code to be converted to 3
```

In this case....(etc).

Fred Gohlke
Carteret, NJ

The staff of Microsystems would like to extend apologies for this error to Mr. Gohlke and to the readers.

The CP/M Connection, Part IV

Dear Editor:

Congratulations on getting a much-needed magazine off the ground and into circulation! Articles on CP/M, such as Chris Terry's "The CP/M Connection," are desperately needed out here in the field, since the Digital Research documentation is often obscure, and books such as Dr. Zaks' recent contribution are written for a rank beginner, in addition to being poorly organized, hastily thrown together, and full of errors.

In Part IV of Mr. Terry's series, there is a discussion on finding the available memory size of a CP/M system. He quite correctly points out that the bottom memory location used by CP/M is contained in the address portion of the jump instructions at locations 0005H-0007H, where it may be read by the user's application program.

There is, however, one small "gotcha" that should be pointed out. One cannot simply crank up DDT and examine these locations to determine the bottom of CP/M, as DDT will change this address to reflect the reduced memory size in the debug mode.

Hence, while an application program can read this location and determine the top of available memory "on the run," you cannot get the correct answer by examining these locations manually with a debugger.

David Goodman
Marshall, VA

IDS-88/Cromemco CPU Fix

Dear Editor

Your article, "S-100 Modems," in the May/June issue gave me the clue I was looking for. I have been trying to get one of the IDS-88 modem boards to work with a Cromemco CPU—with no luck, as mentioned in the article. Spurred on by your note about a fix, I called IDS—about forty times over the last month. I wrote to them and, in desperation, even left a message on their modem phone (one of the most finicky I have ever seen, by the way!). No one is home—ever! One wonders if they are still in business, or how long they will be in business with that level of customer support.

Anyway, with that apparent dead end, I thought I would write to you. Could you please tell me what the fix is, or refer me to someone (a dealer?) who might know? I would appreciate any help you could provide.

While I'm writing—I like your magazine. It fills a void with a level of solid technical detail that is of much interest. Keep up the good work!

Larry Wright
2071 Oakley Ave.
Menlo Park, CA 94025

Editor's Note: *International Data Systems has filed for Chapter XI, and there is some doubt that they will be back in business. We never did discover exactly what the problem was between the IDS-88 Modem and Cromemco CPU cards. If any readers know, would they please let Larry (and* Microsystems*) know.*

COMMX Version 7

Dear Editor:

COMMX version 7 has been enhanced since the *Microsystems* product review by Glenn Hart (May/June 1981). The features added are:

1) IBM file transfer capabilities with TSO + Wilber.
2) The sending of "break" signal for half-duplex systems via control-B.
3) User dial directory for auto-dial PMMI or HAYES modem card types.

ANNOUNCING A REVOLUTION IN THE COST OF PROFESSIONAL SOFTWARE

VISACCOUNT is a fully integrated business and accounting system designed for use in small businesses. VISACCOUNT is extremely comprehensive and professional, yet it is very easy to use. The system is controlled from a series of interconnected menus permitting user-friendly operation. Everything you need to set-up and operate the system is provided with the VISACCOUNT package. Experts have estimated the development costs for a fully integrated software system ranges between $7,200 and $22,000.† When you buy software the developer has to recapture this expense. Computer Services Corporation of America is selling its software with a view that volume sales can almost negate this development cost.

OUR GUARANTEE — Buy both our software and that of our competitors (who will no doubt charge several times our price because they need to recapture their development cost). Compare the two systems and we know you'll return theirs (make sure they'll let you return their software). If you decide not to keep our system, then return it within 45 days for a full refund. Once you've used our system we're confident you'll be delighted.

†Microcomputers for Business, Applications, 1979.

VISACCOUNT

What You Receive

- Nine 5¼'' double density disks (or six 8'' single density disks)
- Easy-to-use operator's manual (over 200 pages)
- Self-study guide on bookkeeping and accounting (over 180 pages)
- Cassette based instruction program on set-up and operation

Available for Apple*, TRS-80, and most others

*The Apple version requires the Microsoft Z80 softcard. CSCA has CBASIC2, CP/M and Microsfot Z80 softcard in stock.

GENERAL LEDGER

Master File
Receipts and Disbursements
Balance sheet
Income Statement
Comprehensive Budget Analysis
Trial Balance and more

ACCOUNTS PAYABLE

Cash Requirements Report
Generates Purchase Orders
Automatic Posting
Prints Checks

INVENTORY MANAGEMENT

Comprehensive Status Reports
Inventory Alert Reports
Raw and Finished Goods
and more

ACCOUNTS RECEIVABLE

Complete Order Entry
Automatic Posting
File Aging (30/60/90 days)
Prints Invoices
Audit Trails

PAYROLL

Comprehensive Payroll Register
Prints Payroll Checks
Automatic Posting
Form 501, W2, 941

FIXED ASSETS ACCOUNTING

Complete Capital Goods Record
Option of 5 Depreciation Methods
IRS Depreciation Report

EXTRA: MAILING LIST PROGRAM

Features

Menu Driven: The entire system runs from a single master menu which accesses numerous subsidiary menus, when needed, to perform the full spectrum of business and accounting functions.

Self-Documenting: All the information needed to use the sytem is provided in an easy to self-study format.

Requirements: 48K CBASIC2
2 DISK DRIVES CP/M

Send $159 for the VISACCOUNT system

COMPUTER SERVICES CORPORATION of AMERICA
332 East 30th Street New York, New York 10016
Order Toll Free 1-800-221-2486 ext. 1591
Technical Number 1-212-685-0090

Name ______________________

Address ______________________

City/State/Zip ______________________

☐ Master Charge ☐ Visa ☐ American Express

No. ______________ Expires ______________

Your System ______________________

Disk Size ☐ 5¼'' double density ☐ 8'' single density

4) Remote control for use with "BYE" or "REMOTE" (PMMI or HAYES) programs.

I feel the following items need clarification regarding software operations and procedures that were misleading in the article:

All versions of COMMX are written entirely in 8080 assembler, allowing a purchaser of the source program to make modifications if desired.

The cost of the COMMX source program ($250) is *not* an obvious attempt to discourage its purchase, as Mr. Hart prematurely concludes. It is directly based on efforts to provide an extensively tested program along with immediate user technical support where professional experience is required. Since every line of source contains a comment, many useful programming techniques will be discovered in the twenty different CP/M function calls implemented. Both async- and synchronous file transfer protocol as well as other extended features are provided to suit the needs of any user. A file transmit option (not mentioned in the article) allows full or no echo wait with a connected host system. Full echo wait indicates echo error, providing a measure of transmit data integrity without any special requirements on the connected host system other than the ability to echo the characters it receives (termed echoplex). The operator can either abort the transfer or note the error location and continue the transfer.

COMMX synchronous protocol is different in concept from others and has major differences! (The article was very misleading on this point.)

• protocol optimization of overhead for reduced online time.

•error-free file transfers using CRC16 polynomial.

•fast CRC16 encode/decode (all registers in 43 bytes!).

•CRC16 formatted for transmission on timeshare ASCII line.

•real-time master/slave operation (XON/XOFF not used here).

• one byte controls consisting of STX, ETX,ACK,NAK,ENQ.

• allows transfer of any data type (COM, REL,INT,BAS,etc.).

COMMX is menu-driven, which means all commands are displayed on the screen and, all input options are prompted, making program operation child's plays (with simple keystrokes). Since most features are automatically selected by the program, it appears to have a reduced command set. Is a program with a larger command set easier to use? I suggest a closer look into the command set options before evaluating all the feature bits in a program's bucket!

COMMX (now version 7) functional documentation is available at no charge from Hawkeye Grafix, 23914 Mobile, Canoga Park, CA 91307.

New features with COMMX version 7 include: transmit break for IBM type half-duplex attention control, file transmit no echo wait line pause and selectable continue character on IBM type half-duplex links, automatic line initialization to 8-bit data no parity (from 7-bit even parity) and console echo switched off with .COM file transfers, duplicate file create warning allows intentional override, auto-dialing modem types accept user created dial directories, auto redial, and remote control capabilities. COMMX still executes in 5K bytes, allowing the rest of available system memory for disk buffer.

Will Pierce
Hawkeye Grafix
Canoga Park, CA

SNOW MICRO SYSTEMS, INC.
P.O. Box 2201 Fairfax, VA 22031
(703) 378-7257

DMA S-100 FRONT PANEL SYSTEM
This front panel system becomes a temporary S-100 bus master to access system memory and I/O, and requires no connection to the CPU card. The front panel system is designed as two cards: a binary switch and display card, and an S-100 interface card. Both cards are 5" x 10". The display card is supplied with a standard "piggy-back" mount to the interface card, but can be separated for chassis mount.

DS.01 kit	$395.00
DS.01 A&T	$495.00

STATION CONTROLLER CARD
The Station Controller was designed for satellite tracking applications. This S-100 card is packed with features like: 7 analog input channels (A to D), 6 reed relays with 1 amp contacts and a 24 hour hardware clock with battery backup provision. Use this card for home energy management, robotics or any real time control application.

SC.01 kit	$245.00
SC.01 A&T	$349.00

RADIO TELETYPEWRITER INTERFACE CARD
The S-100/ RTTY interface that lets your computer talk to the world! Standard baud rates of 45.5, 50, 75 and 110. Mark frequency 2125, space frequency 2195 or 2175, with other frequencies user adjustable.

AR.01 kit	$245.00
AR.01 A&T	$349.00

To order send check or money order. Please include $5.00 (USA) $10.00 (nonUSA). VA residents include 4% tax. Dealer requests invited.

IS YOUR *North Star* OUT OF SORTS?

INCREASE YOUR BASIC'S SORTING POWER OVER 1800%!

N★SORT is easy to use and will perform sorts on one and two dimensional or string arrays using optional sort keys. For example, to alphabetize A$:

```
10 A$ = "ZYXWVUTS"\ REM Define String
20 SRT A$,LEN(A$),1\  REM Sort A$
```

N★SORT interfaces to any release 4 or later North Star Basic and can be yours for ONLY $89 plus $1.50 shipping
Calif. Res. add 6% tax.
Send check VISA or M/C
Complete Brochure Available

SZ Software Systems
1269 Rubio Vista Road, Altadena, Calif. 91001
(213) 791-3202

MIDWEST MICRO WAREHOUSE

3415 Kenwood • Kansas City, MO 64109 • Phone (816) 753-1304

Our ad is so-so, but our pricing is *hot!* Below are a few of the fine products we offer. If you don't find what you're looking for, call us!

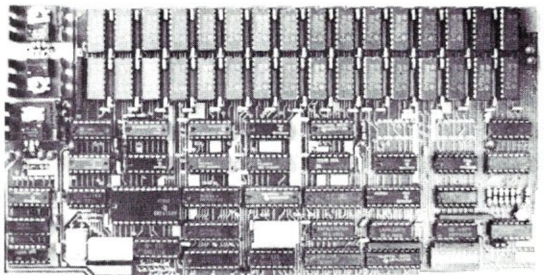
		KIT	A&T
SD Systems	SBC-200	315.	396.
	Expandoran II	195.	337.
	PROM 100	169.	279.
	VF-II	308.	397.
Godbout	Disk 1	N/A	395.
	CPU-Z	195.	230.
	12-Slot Mother	N/A	133.
Micromation	Z-64		1040.
	Doubler		395.
	Multi-User I/O		239.
	System Z		4600.
Seattle	8086 Card Set (W/86-DOS)		655.
	4 Port Serial I/O		240.
	8/16 Static RAM		260.
	16K+ Static RAM		260.
TEI	MCS-112, 122		500./600.
	RM-12, RM-22		630./730.
	DFD-0		520.
	RFD-0		630.

Visual Technology	
Visual 200	900.
Visual 400	1400.
NEC 5510, 5515	2850.
NEC 5520, 5525 (W/Vert. Forms)	3250.
T. I. 810 (Basic)	1695.
(Loaded)	1885.
DataSouth DS-180	1385.
MicroPro Int'l.	
WordStar	315.
MailMerge	110.
DataStar	225.
SuperSort II	135.
Microsoft	
FORTRAN	349.
COBOL	574.
BASIC-80	294.
BASCOM	329.

We ship prepaid or COD certified check, M.O., or cash with 10% deposit. Shipping is from stock to 15 days; most products in stock. And (sorry!), at these prices, we cannot pay shipping charges.

256K RAM IN 4K BLOCKS

1/4 Megabyte $1495



OVERVIEW

The BSR 64/256 is an 8 bit bank selectable dynamic random access memory card designed to operate in a Z-80 based S-100 computer system with a CPU clock frequency of up to 4 MHz(A model) or 6 MHz(B model).

Individual 64K banks are selected via the IEEE 696 8 bit address us extension. If the host system is not capable of driving the extended address bus, one of the BSR 64/256 cards in the system may be configured to drive it through an onboard latched output port.

System area is allocated in 4K blocks by writing a system mask out to two latched output ports. Another port allows any one of up to eight cards to be assigned as the current system master. Logically, up to 64 cards may be addressed in a single computer system.

Assembled & Tested Boards

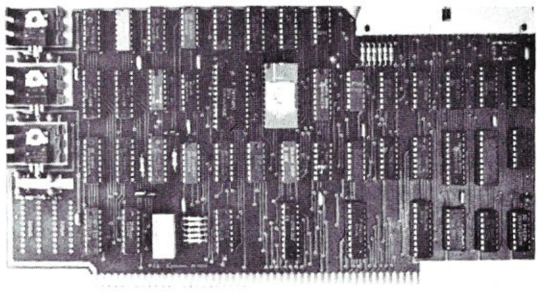
(4164)	64k	$770
	128k	$1015
	192k	$1260
	256k	$1495

Memory prices fluctuate, call for price

FEATURES

MAX STORAGE	256k bytes (32 ea. 64k X 1 chips) or 64k bytes (32 ea. 16k X 1 chips)
BLOCK SELECTION	Any combination of 4k blocks in any 64k bank, software selectable
SYSTEM AREA	Any combination of 4k blocks in low order bank of current system master card, software selectable
SYSTEM MASTERS	Any one of up to 8 cards software selectable, one card jumper selectable for system power-up or reset
CARDS/SYSTEM	Logically, up to 64 64k or 256k cards
BANK SELECTION	Uses or implements IEEE 696 (S-100) extended address bus
CPU SPEED	4 MHz or 6 MHz with no wait states
REFRESH MODE	Invisible
OUTPUT PORTS	4 Consecutive ports for entire system, selectable on any 4 port boundary

14″ WINCHESTER FIXED DISK CONTROLLER & SUBSYSTEM



FDC 4000

14″ Winchester Fixed Disk Controller

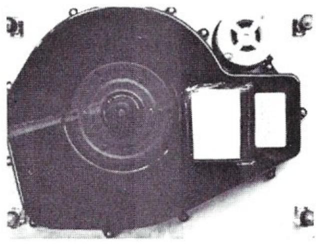
Controls Shugart 4000 & 4100 series drives
Up to 174 megabytes per controller
I/O mapped buffer - no wait states
Includes MP/M* XIOS & CP/M* CBIOS

Assembled & tested $895

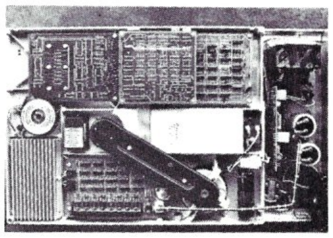
29 Megabyte Disk Subsystem $3840

The fixed disk subsystem includes a 29 million byte Shugart fixed disk (SA 4008), and an assembled and tested S-100 disk controller (FDC 4000). Also included is an unfinished rack mountable aluminum case, punched and ready for assembly. To complete this kit we have included power supply, fan, wiring harness, signal cable, and MP/M* XIOS and CP/M* CBIOS.

(SA 4008)



(Assembled Subsystem)



PCC SYSTEMS

"Contact Us For More Information"

5232 Manzanita Ave. Carmichael Ca 95608 (916) 338-5454

Price subject to change

*Trademark of Digital Research

DISK DRIVE WOES? PRINTER INTERACTION?
MEMORY LOSS? ERRATIC OPERATION?

Don't Blame The Software!

Power Line Spikes, Surges & Hash could be the culprit! Floppies, printers, memory & processor often interact! Our unique ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash.

Clear up Software and System problems with an ISOLATOR!

ALL ISOLATORS: • 125 VAC, Standard 3-prong plug
• 1875 W MAX Loabd — 1 KW/Socket or socket bank
• Balanced Pi Filtered sockets or socket banks
• Spike/Surge Suppression — 1000 Amps, 8/20 usec
(SUPER ISOLATORS offer expanded filtering and Spike/Surge Suppression capabilities)

ISO-1

ISO-2

ISO-1	-3 individually filtered sockets	$ 62.95
ISO-4	-6 individually filtered sockets	106.95
ISO-2	-2 filtered banks; 6 sockets	62.95
ISO-5	-3 filtered banks; 9 sockets	87.95

***SWITCHABLE ISOLATORS —**
ALL ISOLATOR advantages combined with the versatility, convenience and utility of individually switched sockets. Each switch has associated pilot lite.

ISO-6

ISO-6	-3 switched, filtered sockets	$141.95
ISO-8	-5 switched, filtered sockets	178.95
ISO-3	-3 super filtered sockets	94.95
ISO-7	-5 super filtered sockets	154.95

***SUPER ISOLATORS** — Cure for severe interference problems. Useful for Industrial applications and heavy duty controlled equipment or peripherals.

ISO-7

• Dual Balanced Pi Filtered sockets
• Spike/Surge Suppression — 2000 Amps, 8/20 usec

***CIRCUIT BREAKER any model (add-CB)**	**ADD 8.00**
***CKT BKR/SWITCH/PILOT any model (CBS)**	**ADD 16.00**

Master-Charge, Visa, American Express
TOLL FREE ORDER DESK 1-800-225-4876
(except AK, HI, MA, PR & Canada)

ESP Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760
Technical & Non-800 1-617-655-1532

The CP/M* Bus

by Anthony Skjellum

The topic of discussion this month is once again enhancements for CP/M2. Due to space limitations, the section on the internals of link records will have to be deferred to the next installment.

Sub-directories

Operating systems on large computer systems usually provide sophisticated filing systems which allow sub-directory structures to be established within a user's main directory. This allows various files to be collected in a logical way. CP/M2 provides no such named sub-directories, although it provides the USER feature which allows files to be separated into sixteen categories on a given disk. Hard disk and high density floppy systems are making large amounts of on-line storage available on CP/M systems. However, greater storage capacity leads to problems in data organization which can be alleviated by the use of sub-directories.

At the outset of this series of enhancements for CP/M, we stated that a major premise of enhancements would be to stay within the original architecture of the software system. Therefore, in discussing sub-directories, we will propose a conservative approach.

The above consideration limits our choices for implementing sub-directories, but does not remove the possibility of making a powerful but feasible expansion to CP/M. To provide sub-directories, we will define a new variety of file to be known as a directory file. Directory files will have the file type of .DIR by default; but this will not be required. A status bit (denoted as b4') will be used to indicate that the file is a directory. Directory files will reside in any of the sixteen user areas, but will not nest. Such a file will contain a fixed number (e.g. 64) of 32 byte entries in the same way the CP/M maintains its main directory. To a CP/M system, it will appear outwardly as a single file and the directory file will allocate all the disk space of its constituents. Thus, it will be possible to treat it as a whole or as a set of files for the sake of various CP/M operations.

Several new BDOS commands and CCP functions will be needed to provide the sub-directory feature. We will begin by describing the new BDOS commands needed.

To begin with, we will need a new BDOS command which will create the directory file. The command will be known as MKDIR. It will set the directory status bit b4' to indicate that the file is a directory. Furthermore, the A register will be passed with the number of directory entries to set up for the directory, and sufficient storage will be allocated to contain these directory entries. Also, the first record of the directory file will contain information including the number of entries which the directory can contain. Here is an example of a typical BDOS call using MKDIR:

```
MKDIR <dir>  ; <dir> is an unambiguous file name
```

This example creates a ten entry directory from the file name stored in the FCB marked "fcb." As shown in the above example, the accumulator is returned with zero for a successful operation. The value 1 will be returned for a file exists condition, and -1 for a disk write error condition.

Now that we have defined the way to make sub-directories, there must also be a BDOS command to change CP/M's working directory from the main directory to a sub-directory. It must also be possible to change back to the main directory when necessary. This command will be denoted as CD (which stands for change directory, as in Unix). When calling CD, the DE register will point to an FCB with the file name of the directory to change over to. Upon return, the accumulator will contain zero if the operation was successful. A value of -1 will indicate that the file did not exist, and 1 indicates that the file was not a directory. Also, passing a value of zero in DE will cause CP/M to return to the main directory.

Once the working directory has been changed to a subdirectory, the search first and search next functions will examine the sub-directory and operation will be totally transparent to transient programs. Furthermore, when the simple link facility provided by LN is used within a sub-directory, it will refer back to the file in the current main directory and not in user area zero. Also, sub-directories in user area zero will be permitted to have simple link files.

The BDOS file delete command will also remove files from a sub-directory in the expected way. However, it will be very simple to delete the directories themselves when in the main directory. Therefore, caution would have to be exercised. Certainly the CCP ERA command would have to be changed to confirm any directory deletion.

Besides the above commands, it will also be necessary to generalize the rename capability of CP/M to allow renaming of files between sub-directories, and between a sub-directory and the main directory. This will be provided by a single BDOS command called MV, again using the Unix convention for naming. On entry, the DE pair will point to an FCB with the old and new names for the file as in the standard rename command (see p. 20 of *CP/M 2.0 Interface Guide*). However, both the accumulator (A) and the HL register pair will be used to pass further information. Calling with the accumulator set to zero will indicate that we are requesting a directory-to-directory rename. In this case, HL must point to an FCB with the name of the source directory followed by the destination directory in the second 16 bytes of the FCB. This format is just like that used for the file names themselves pointed to by DE. Finally, the accumulator is returned with the same error codes as for a standard rename.

If the accumulator is non-zero, the BDOS will expect a transfer between a sub-directory and the main directory. The HL register will point to an FCB containing the name of the sub-directory. If A is odd, (e.g. 00000011B), then the transfer is made to the main directory. Otherwise the transfer will be made to the sub-directory pointed to by HL.

Certain shorthands will also be allowed within the FCB. A ^A as the first character of the file name will indicate the current directory, while a ^B will indicate the main directory. Also, return current directory name command will be supported. This command will return a pointer to the directory name so that transients can determine the sub-directory they are in. The null string is returned for the main directory. This command will also be used by a new CCP function called PWD (again after Unix) which will display the name of the working directory. For the main directory, PWD will display nothing.

MULLEN

S-100 PRODUCTS

1 EXTENDER BOARD WITH LOGIC PROBE **$89 assm/tested**
with these features for use in testing your S-100 boards.

- Logic probe with display shows; (H) for TTL logic high, (L) for low, (O) for open or 3-state, and (P) for pulse.
- Pulse catcher switch latches (P) aids in detecting infrequent pulses.
- Jumper links in +8 and ±16 volt lines allow current measurement, switching and fusing.
- Interlaced signal and ground traces reduce noise.
- Pushbutton reset allows restarting test programs.
- Formed leads on both sides of the edge connector for easy scope probe attachment.
- Prototyping area and regulated 5 volts allows construction of special test circuits on the board.
- Edge connector label shows signal names and pin numbers.
- 5 1/2" high, on quality FR-4 material, solder masked and gold plated on mating surfaces.

2 INDUSTRIAL EXTENDER BOARD **$99 assm/tested**
saves time where many boards are tested every day.

- ZERO-INSERTION FORCE edge connector.
- Switch and indicator light control +8 and ±16 volt power.
- Pushbutton reset allows restarting test programs.
- Fuses in power lines protect test computer.
- Interlaced signal and ground traces reduce noise.
- Formed leads on both sides of edge connector for easy scope probe attachment.
- Edge connector labels show signal names and pin numbers
- 6" high, on quality FR-4 material, solder masked and gold plated on mating surfaces.

3 RELAY OPTO-ISOLATOR CONTROLLER BOARD **$219 assm/tested**
for signal switching, or controlling low power devices.

- 8 reed relays.
- 8 opto-isolators with input bridge rectifiers, series resistors, and filter capacitors.
- 256 switch selectable port addresses.
- Removable terminal block for use with up to 16 AWG wire.
- LED indicators in relay drive circuits.
- Socket for input simulation or testing.
- Quality FR-4 material, solder masked & gold plated on bus connector.
- Instructions include programming examples.

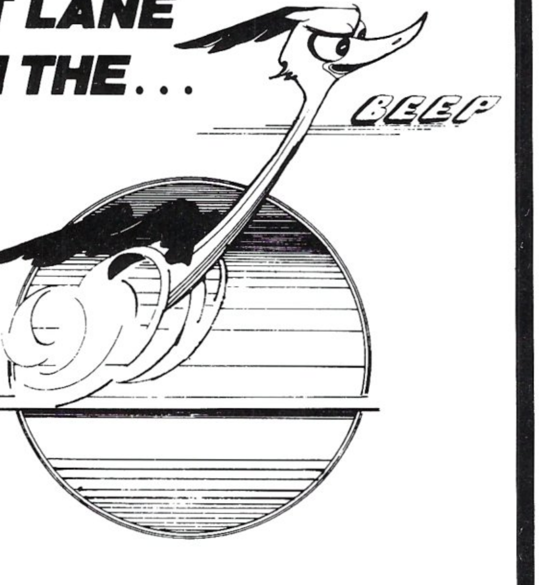
4 TRIAC OPTO-ISOLATOR CONTROLLER BOARD **$219 assm/tested**
for controlling line voltage AC devices.

- 8 triacs with snubbers for controlling inductive loads, and zero crossing isolated drive circuitry.
- 8 opto-isolators with input bridge rectifiers, series resistors, and filter capacitors.
- 256 switch selectable port addresses.
- Removable terminal block for use with up to 16 AWG wire.
- LED indicators in triac drive circuits.
- Socket for input simulation or testing.
- Quality FR-4 material, solder masked & gold plated on bus connector.
- Instructions include programming examples.

MULLEN COMPUTER PRODUCTS
BOX 6214, HAYWARD, CA 94544

OR PHONE (415) 783-2866 · VISA/MASTERCHARGE ACCEPTED.
INCLUDE $1.50 FOR SHIPPING & HANDLING.
CALIFORNIA RESIDENTS ADD TAX.

MOVE INTO THE FAST LANE WITH THE . . .



AZTEC C COMPILER!

☐ ALL C LANGUAGE FEATURES EXCEPT FLOAT, DOUBLE, AND LONG DATA-TYPES AND BIT FIELDS

☐ STRICT ADHERENCE TO THE DEFINITION OF C IN THE *C PROGRAMMING LANGUAGE* BOOK BY KERNIGHAN AND RITCHIE

☐ EXTENDED LIBRARY ROUTINES FOR CP/M INTERFACING AND OTHER SPECIALIZED FUNCTIONS

☐ SOME SPECIFICS:

- short and unsigned datatype
- EXTRNS are implemented as defined by K & R to provide flexibility and ease of use in implementing modular systems
- structures and unions (version 7 extensions available as a compile time option)
- while, for, do/while, switch/case, and go to
- multidimensional arrays
- conditional compilation with:
  #ifdef #ifndef #else #endif
- all C operators
- declarations of complex datatypes
- command line arguments using argv and argc
- static, initialized, and register variables
- all other C language features except float, double and long datatypes and bit fields

☐ RUN TIME LIBRARY IS SUPPLIED IN SOURCE FORMAT AND INCLUDES: fopen fclose getc putc iseek open close printf read write and more

☐ FAST EXECUTION AND COMPILATION TIMES (EXECUTES THE 9/81 BYTE BENCHMARK IN 25 SECONDS — FASTER THAN ALL C COMPILERS EXCEPT WHITESMITHS C)

☐ THE COMPILER PRODUCES ASSEMBLY LANGUAGE SOURCE THAT IS (AS A COMPILE TIME OPTION) COMPATIBLE WITH MICROSOFT'S MACRO-80

☐ PACKAGE INCLUDES RELOCATING ASSEMBLER AND LINKAGE EDITOR

☐ FLOATS AND LONGS AVAILABLE IN LATE '81

AZTEC C, ASSEMBLER, AND LINKER	$135
WITH K & R BOOK	$150
AZTEC C UNIX CROSS COMPILER (PDP 11)	$850

TECHNICAL SOFTWARE SYSTEMS
BOX 55, SHREWSBURY, N. J. 07701
(201) 780-4004

N. J. residents add 5% sales tax — VISA and MASTERCARD

AZTEC C is a trademark of MANX Software Systems
CP/M is a trademark of Digital Research
UNIX is a trademark of Bell Labs

CP/M Bus, continued...

To complement the new BDOS commands are several other new CCP commands. The command MKDIR will make a directory and has the following syntax:

```
MVI     C,MKDIR ; command is mkdir
LXI     D,FCB   ; point to fcb with name of file
MVI     A,10    ; 10 entries for this directory
CALL    BDOS    ; execute call
ORA     A       ; see if error...
JNZ     ERROR   ; yes...
```

Also, a change directory command (CD) will be provided. It will take two forms:

```
CD <dir> ; change to a directory
CD       ; change back to main directory
```

Similarly, the REN command will support renaming between sub- directories and the main directory. By specifying a directory name in square brackets ([]) before the file name, REN will understand that it is to do an inter-directory move. Specifying [] indicates the main directory. An example is presented here for clarity:

```
CD TEMP                      ; change to directory TEMP.DIR
REN TEST.TXT=[]TEST.TXT      ; get TEST.TXT from main
REN [TEMP2]TEST.TXT=TEST.TXT ; move TEST.TXT from
                             ; TEMP.DIR to TEMP2.DIR.
```

Finally, the DIR command will be generalized to list the contents of directory files when the second argument D is specified.

One further CCP command will be needed. This will be the PATH command. The PATH command will indicate to CP/M which sub- directories it should search for .COM files after looking in the current directory. A BDOS command will also be provided. The BDOS command will allow a transient to give CP/M the address of a set of "half"-FCB's containing the names of the path for the logged disk. This will remain in effect until a cold boot. A transient will also be able to pick up the path set before it was invoked. This command will return an address vector to a set of "half"-FCB's containing the information.

The CCP PATH command mentioned above takes two or more arguments. The first argument is either a plus sign, a minus sign or a disk name. Plus indicates that the following list of directories are to be added to the path list, while a minus sign removes the list from the path. An alternate form of the command reports the current path of a disk. This is done by providing a disk name instead of the plus or minus sign. Examples follow:

```
PATH+ ASM1 ASM2 TEXT1 ;add ASM1.DIR, ASM2.DIR
                      ; and TEXT1.DIR to path
PATH - TEXT4.DOC      ; remove TEXT4.DOC
                      ; directory from path.
PATH B:               ; report current path of B:
```

The path feature should further reduce the need for extra copies of standard programs and utilities.

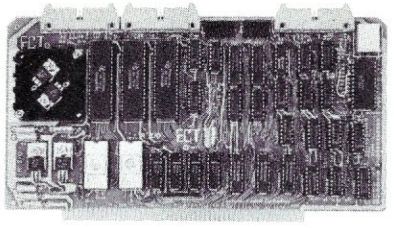
Some additions to PIP are also in order. In following the CP/M2 convention for handling user areas, we will allow PIP to copy into the current directory, but not into other directories. This feature will be provided by generalizing the G option. Currently, G is followed by the user area number. We now add the option of adding a colon and the subdirectory name (terminated by Z) to this specification. Thus, we might do something like the following:

```
PIP TEST.FIL=TEST.FIL[G0:SUBDIR^Z] ; this will get TEST.FIL
                                   ; from the sub-directory
                                   ; SUBDIR.DIR in user area
                                   ; zero.
```

The topic for the next installment will be a discussion of the internals of link records used in complex link files. ■

CUSTOM PRODUCTS

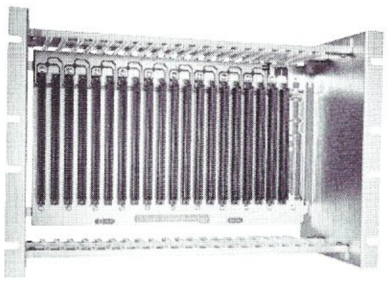
DESIGN • LAYOUT MANUFACTURING



R ²I/O

ROM/RAM & I/O

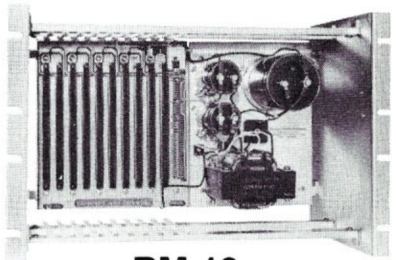
S-100 PRODUCTS



ECT-100-F

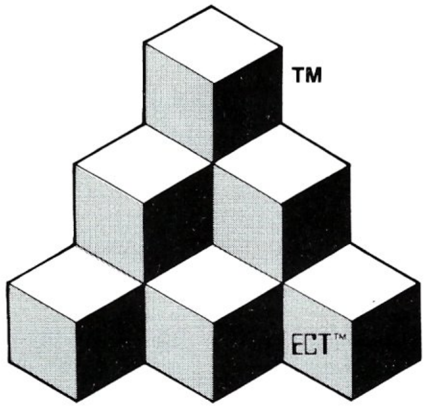
RACKMOUNT CARD CAGES

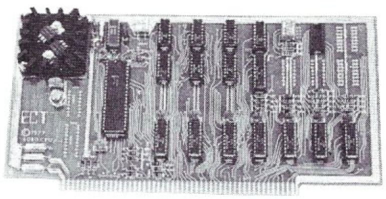
ECT™



RM-10

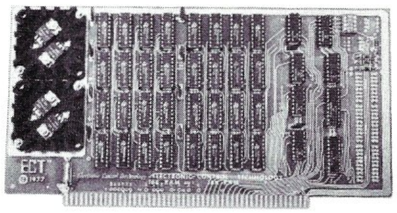
CARD CAGE & POWER SUPPLY





8080 CPU

CENTRAL PROCESSING UNITS

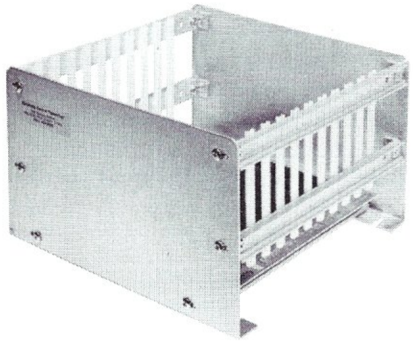


16K RAM

FULLY STATIC MEMORY

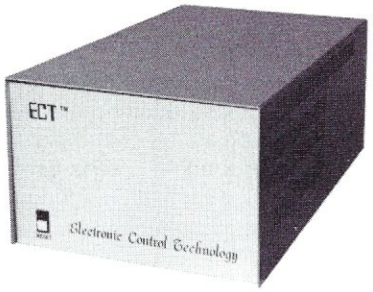
BUILDING BLOCKS FOR MICROCOMPUTER SYSTEMS, DEDICATED CONTROLLERS AND TEST EQUIPMENT

CARD CAGES, POWER SUPPLIES MAINFRAMES, CPU'S, MEMORY I/O, OEM VARIATIONS



CCMB-10-F MIN

6,10 OR 20 SLOT CARD CAGES



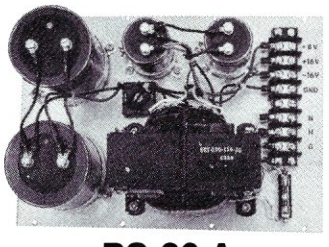
TT-10

TABLE TOP MAINFRAMES

ELECTRONIC CONTROL TECHNOLOGY, INC.

763 Ramsey Ave., Hillside, NJ 07205 (201) 686-8080

SPECIALIZING IN QUALITY MICRO COMPUTER HARDWARE



PS-30 A

POWER SUPPLIES

MULTIBUS® PRODUCTS

MULTIBUS IS A TRADEMARK OF INTEL CORP.

An Introduction To The C Programming Language

by David A. Gewirtz

An introduction to the C language and reviews of four popular microcomputer implementations

C is the language that was developed after B. Although its name isn't very original, the C language is a highly versatile programming tool. C cannot be classified either as a "higher" or a "lower" level language. Probably the best classification would be a low-level, higher level language. It has also been called a systems implementation language.

C has many of the higher level control constructs, such as while loops, if-else conditionals, and block structuring. C also easily manipulates machine-related data types such as the byte with operations like bit-wise shift This combination makes C very useful for programming operating systems, languages, and utilities. It can also be easily used for other applications as well.

C, originally developed at Bell Laboratories, has evolved from the BCPL language created by Martin Richards, and the B language written by Ken Thompson for the first UNIX operating system. It shares many of the features of both BCPL and B, although where they concentrated almost exclusively on machine words, C has been expanded to include larger integers, characters, and in some implementations, floating point numbers. In addition, the syntax is different between BCPL and C. C also borrows many ideas, like the typing of variables, from other languages including ALGOL, Pascal, and PL/I. The most noted application of C was the development of the UNIX operating system for the PDP-11 computer.

Unlike Basic, C is a compiler, not an interpreter. This means that once compiled, programs run much faster because the statements in the program do not have to be retranslated into machine code each time they are encountered, which can become a very time consuming operation. In addition, the resulting object program that is run takes up much less space because the interpreter does not have to be resident in memory while the program is running.

David A. Gewirtz, 38-67 Taylor Rd., Fairlawn, NJ 07410.

C does not have the line numbers used in most versions of Basic. Instead, a C program is organized into manageable routines called functions. These functions, combined with the ability to make compound statements, make programs very straightforward. Wherever a simple statement can be used, a compound statement can be used there instead. For example, in the following "b=9" is a simple statement:

```
  if (a==5) b=9;
```

Any group of statements, enclosed in brackets "{ " and " }", can be substituted for that simple statement:

```
  if (a==5)
        {
        b=9;
        c=36;
        if (count<8) ++count;
        }
```

The operators "==" and "++" will be explained later.

These compound statements have a recursive definition. A compound statement is a statement containing other statements. Any of the statements inside a compound statement can also be a compound statement. This very powerful way of using program control statements allows programs to be "block-structured." I will not argue the relative merits of structuring, save that it makes programs easier to design, and clearer to debug and understand. This same block-structuring construct can be found in ALGOL and Pascal.

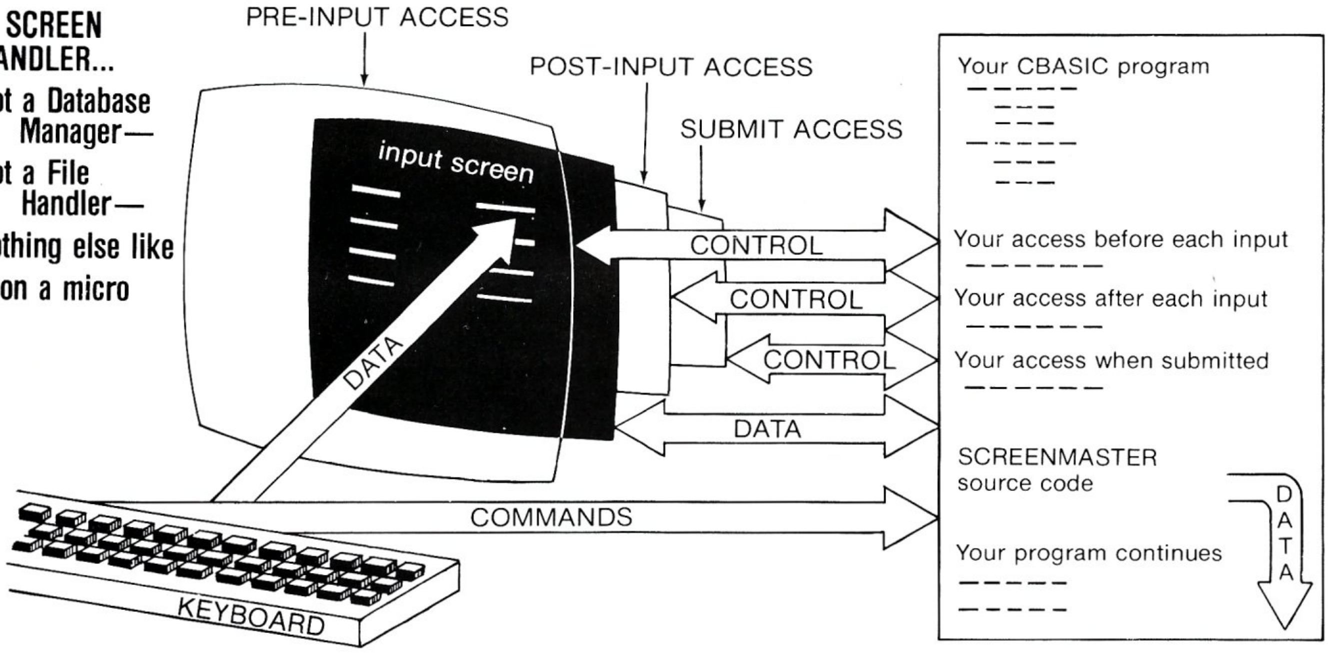
Functions are usually small, stand-alone subprogram segments of the main program. They are similar to the

At last! A "SCREEN HANDLER" for CBASIC2 programmers.
...source code to handle full screen input with multiple screens.

SCREENMASTER®

a program module that makes data entry a snap.

A SCREEN HANDLER...
not a Database Manager—
not a File Handler—
nothing else like it on a micro



- %include directly in program
- full screen input
- multiple screens
- user commands
- total programmer control in accesses
- terminal independent (>= 80 x 24)
- default editing with overriding
- screen maker utility
- screen tester utility
- run time terminal configurator utility
- separated programmer interface module
- easy to use • easy to install
- demo disk available
- extensive manual
- not a database manager or file handler
- runs with CP/M and CBASIC2

%INCLUDE DIRECTLY IN PROGRAM. SCREENMASTER is provided in source code and can be included directly in your program as if you had written it yourself. Total control of your program is thus assured.

FULL SCREEN INPUT WITH MULTIPLE SCREENS. User enters data on the formatted screen of your design. A single screen can be converted to multiple screens without program changes.

USER COMMANDS. User has commands like FORWARD (n fields), BACKWARD (n fields), GOTO (field m), NEXT (screen), PRIOR (screen), or PRINT (screen p). User commands may be abbreviated, implemented as special function keys, or individually disabled.

**CP/M a trademark of Digital Research, Inc. CBASIC2 a trademark of Compiler Systems, Inc.*

TOTAL PROGRAMMER CONTROL IN ACCESSES. Programmer is given access to SCREENMASTER operations in the exits—pre-input, post-input, and submit. The post-input exit, for example, lets you override SCREENMASTER editing, change any field, issue any command, and issue or override any return code.

UTILITIES INCLUDED. The SCREEN MAKER utility lets you develop and view your screen design interactively. The SCREEN TESTER utility lets you simulate the workings of your program by giving you interactive control in the exits. The TERMINAL CONFIGURATOR utility writes your terminal characteristics to a file that is read in when your program runs. Your terminal can be easily added to the list of available terminals.

EASY TO USE. Even a novice programmer can use it. Issue a simple GOSUB. When it returns your validated data is available for your further use. No programming is necessary. But if you need to do something special—like search a file—you can do it yourself in the exits.

FLEXIBLE. If you're an experienced programmer you'll appreciate the lengths we've gone to to enable you to interface and modify SCREENMASTER. Initialization is a separate, well documented module. All communication with your data and screen formats is via functions in another separate, well documented module. Even the guts of SCREENMASTER is well documented, allowing some simple modifications.

Send your check for $195 to **MARKETING ESSENTIALS, INC.**, 206 Mosher Avenue, Woodmere, N.Y. 11598. Or call (516) 569-4533/(800) 645-2622. Manual available separately for $25, applicable towards the full purchase price. A demo disk is available for an additional $10.

MARKETING ESSENTIALS, INC.
206 Mosher Avenue/Woodmere, N.Y. 11598   Telephone (516) 569-4533/(800) 645-2622

Basic gosub statement in that when they are called, control jumps to them, and at the end they can return to where they came from. That is the only similarity. In Basic, all variables are global. If a routine to position the cursor on a terminal at X and Y coordinates was used in Basic, it might look like this:

```
330 REM Position Cursor at A and B
340 X=A
350 Y=B
360 GOSUB 5000

4999 REM Direct Cursor Addressing
5000 PRINT CHR$(27)+"="+CHR$(32+Y)
     +CHR$(32+X);
5010 RETURN
```

You would not be able to use X and Y for anything but that routine, and would have to assign the actual values A and B each time the routine had to be called. In addition, you would have to remember that 5000 was the cursoring routine. The same thing in C would be the call:

```
cursor(a,b);
```

which is much clearer to understand. The function would be:

```
cursor(x,y)
     {
     outchr(27);
     outchr('=');
     outchr(32+y);
     outchr(32+x);
     }
```

Outchr is another function which puts one byte to the screen. Once the cursor function is known to be working, it can be put aside and ignored, save to remember to use the function cursor(x,y) whenever it might be needed.

C makes very extensive use of the function feature. The C compiler can be relatively small in terms of reserved compiler operations, and all extraneous operations can be custom designed to fit a user's needs. In fact, C does not have any defined I/O, rather it consists of machine language functions to be called when needed. With functions, programs are very simple to modify. For example, if a new terminal were to be used, all that would be required would be a change in the function. The program remains the same.

Most C compilers come with a run-time library that consists of many previously compiled standardized functions such as outchr and cursor. The programmer simply has to remember the rules of what goes in and comes out of each function, but doesn't even have to include them in the actual program file. They simply must be "linked" to the compiled program before it can run.

The flow of control throughout C programs also helps to promote the readability and structure of the programs. C has many statements that evaluate conditions and control program flow accordingly.

if (expression) < statement > [else < statement >]

If a given condition is true, do something (a statement which can always be a compound statement), else (if it is not true) do something else. If the else is omitted, flow will continue directly after the statement with the if.

while (expression) < statement >

While a given condition is true, do something until that condition is no longer true.

do < statement > while (expression)

Do something while a given condition is true. This differs from the while in that the condition is checked after running through it, not before.

for (expr1; expr2; expr3) < statement >

This works somewhat like the Basic FOR statement. It controls looping through a statement by evaluating three expressions. The first expression is evaluated once, then as long as expr2 evaluates to something other than zero, the statement is executed and expr3 is evaluated. This could be seen as:

```
for (i=n ; i==j/3 ; ++i) <statement>
```

which says "for i becomes n, until i equals j/3, do the statement, and increment i." The symbol "++" is an increment operator which will be examined later.

goto < identifier >

Branch to the label specified by < identifier >.

switch (expression) < statement >

This can be seen as a very powerful ON-GOTO statement. It evaluates the expression, and if it matches any of the cases in the statement, it will execute from that point. A break statement will return control outside of the block. For example:

```
printf("What is your choice?");
getchr(choice);
```

This will get the choice for some set of options. Toupper is a function to convert lowercase alphabetic characters to their uppercase equivalent. If "B" was chosen, the function called reboot would be executed. When it finished, the break would be encountered which would return control outside the switch. If "Z" were chosen, the screen would be cleared, and if none of the options listed were chosen, the error message would be printed.

In addition to the statements above, C has a few very powerful operators. Unary operators are operators that do an operation to a single piece of data. The following is a list of the C unary operators. All of the operators return a value to some expression. For example, in Basic, -X does not mean negate X, but rather returns the negative of X to a specific expression.

```
*p - Pointer to nnn.
     The value is the value of the object currently
     pointed to by nnn.

&x - Address of x.
     Returns a pointer to x.

+x - States that x is positive.
     Returns the same value of x.

-x - Negative of x.
     Returns the negative of x.

++x - Increment x.
      The result is the new value of x.

--x - Decrement x.
      The result is the new value of x.

x++ - Increment x.
      The result is the original value of x.

x-- - Decrement x.
      The result is the original value of x.
```

NEW! TPM* for TRS-80 Model II
NEW! System/6 Package
Computer Design Labs

Z80* Disk Software

We have acquired the rights to all TDL software (& hardware). TDL software has long had the reputation of being the best in the industry. Computer Design Labs will continue to maintain, evolve and add to this superior line of quality software.

— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

All of the software below is available on any of the following media for operation with a Z80 CPU using the CP/M* or similar type disk operating system (such as our own TPM*).

for TRS-80* CP/M (Model I or II)
for 8" CP/M (soft sectored single density)
for 5¼" CP/M (soft sectored single density)
for 5¼" North Star CP/M (single density)
for 5¼" North Star CP/M (double density)

BASIC I

A powerful and fast Z80 Basic interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADGO for "chaining", COPY to move text, EXCHANGE, KILL, LINE INPUT, error intercept, sequential file handling in both ASCII and binary formats, and much, much more. It runs in a little over 12 K. An excellent choice for games since the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II

Basic I but with 12 digit precision to make its power available to the business world with only a slight sacrifice in speed. Still runs faster than most other Basics (even those with much less precision). $99.95/$15.

BUSINESS BASIC

The most powerful Basic for business applications. It adds to Basic II with random or sequential disk files in either fixed or variable record lengths, simultaneous access to multiple disk files, PRIVACY command to prohibit user access to source code, global editing, added math functions, and disk file maintenance capability without leaving Basic (list, rename, or delete). $179.95/$25.

ZEDIT

A character oriented text editor with 26 commands and "macro" capability for stringing multiple commands together. Included are a complete array of character move, add, delete, and display function. $49.95./$15.

ZTEL

Z80 Text Editing Language - Not just a text editor. Actually a language which allows you to edit text and also write, save, and recall programs which manipulate text. Commands include conditional branching, subroutine calls, iteration, block move, expression evaluation, and much more. Contains 36 value registers and 10 text registers. Be creative! Manipulate text with commands you write using Ztel. $79.95/$25.

TOP

A Z80 Text Output Processor which will do text formatting for manuals, documents, and other word processing jobs. Works with any text editor. Does justification, page numbering and headings, spacing, centering, and much more! $79.95/$25.

MACRO I

A macro assembler which will generate relocateable or absolute code for the 8080 or Z80 using standard Intel mnemonics plus TDL/Z80 extensions. Functions include 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical operations, local and global symbols, chaining files, linking capability with optional linker, and recursive/reiterative macros. This assembler is so powerful you'll think it is doing all the work for you. It actually makes assembly language programming much less of an effort and more creative. $79.95/$20.

MACRO II

Expands upon Macro I's linking capability (which is useful but somewhat limited) thereby being able to take full advantage of the optional Linker. Also a time and date function has been added and the listing capability improved. $99.95/$25.

LINKER

How many times have you written the same subroutine in each new program? Top notch professional programmers compile a library of these subroutines and use a Linker to tie them together at assembly time. Development time is thus drastically reduced and becomes comparable to writing in a high level language but with all the speed of assembly language. So, get the new CDL Linker and start writing programs in a fraction of the time it took before. Linker is compatible with Macro I & II as well as TDL/Xitan assemblers version 2.0 or later. $79.95/$20.

DEBUG I

Many programmers give up on writing in assembly language even though they know their programs would be faster and more powerful. To them assembly language seems difficult to understand and follow, as well as being a nightmare to debug. Well, not with proper tools like Debug I. With Debug I you can easily follow the flow of any Z80 or 8080 program. Trace the program one step at a time or 10 steps or whatever you like. At each step you will be able to see the instruction executed and what it did. If desired, modifications can then be made before continuing. It's all under your control. You can even skip displaying a subroutine call and up to seven breakpoints can be set during execution. Use of Debug I can pay for itself many times over by saving you valuable debugging time. $79.95/$20.

DEBUG II

This is an expanded debugger which has all of the features of Debug I plus many more. You can "trap" (i.e. trace a program until a set of register, flag, and/or memory conditions occur). Also, instructions may be entered and executed immediately. This makes it easy to learn new instructions by examining registers/memory before and after. And a RADIX function allows changing between ASCII, binary, decimal, hex, octal, signed decimal, or split octal. All these features and more add up to give you a very powerful development tool. Both Debug I and II must run on a Z80 but will debug both Z80 and 8080 code. $99.95/$20.

ZAPPLE

A Z80 executive and debug monitor. Capable of search, ASCII put and display, read and write to I/O ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format tape, and more! on disk $34.95/$15.

APPLE

8080 version of Zapple $34.95/$15.

NEW! TPM now available for TRS-80 Model II!

TPM*

A NEW Z80 disk operation system! This is not CP/M*. It's better! You can still run any program which runs with CP/M* but unlike CP/M* this operating system was written specifically for the Z80* and takes full advantage of its extra powerful instruction set. In other words its not warmed over 8080 code! Available for TRS-80* (Model I or II). Tarbell, Xitan DDDC, SD Sales "VERSA-FLOPPY", North Star (SD&DD), and Digital (Micro) Systems. $79.95/$25.

SYSTEM MONITOR BOARD (SMB II)

A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1200/2400 baud cassette tape interface, sockets for 2K of RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Bare board $49.95/$20.

ROM FOR SMB II

2KX8 masked ROM of Zapple monitor. Includes source listing $34.95/$15.

PAYROLL (source code only)

The Osborne package. Requires C Basic 2.
5" disks $124.95 (manual not included)
8" disks $ 99.95 (manual not included)
Manual $20.00

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

By Osborne, Requires C Basic 2
5" disks $124.95 (manual not included)
8" $99.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)

By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/6

TPM with utilities, Basic I interpreter, Basic E compiler, Macro I assembler, Debug I debugger, and ZEDIT text editor.
Above purchased separately costs $339.75
Special introductory offer: Only $179.75 with coupon!!

$160.

This Coupon Is Worth One Hundred And Sixty Dollars Toward The Full Price Of The SYSTEM/6 Package System/6 with this coupon is only $179.95. This is a limited time offer.

$160.00

ORDERING INFORMATION

Visa, Master Charge and C.O.D. O.K. To order call or write with the following information.

1. Name of Product (e.g. Macro I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D.) include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS 80, Tarbell, Xitan DDDC, SD Sales (5¼" or 8"). ICOM (5¼" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

Manual cost applicable against price of subsequent software purchase in any item except for the Osborne software.

For information and tech queries call
609-599-2146

For phone orders ONLY call toll free
1-800-327-9191
Ext. 676
(Except Florida)

OEMS

Many CDL products are available for licensing to OEMs. Write to Carl Galletti with your requirements.

* Z80 is a trademark of Zilog
* TRS-80 is a trademark for Radio Shack
* TPM is a trademark of Computer Design Labs. It is not CP/M*
* CP/M is a trademark of Digital Research
Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER DESIGN LABS

342 Columbus Avenue
Trenton, N.J. 08629

```
 x - Returns the ones complement of x.

!x - Not x.
     The result is 1 if x is 0, otherwise 0.

(type-name)x - coerce x to the type type-name.
     Returns the value obtained by converting the
     value of x to that type.

sizeof x - The result is an integer value equal
     to the size in bytes of x.

sizeof (type-name) - The result is an integer value
     equal to the size in bytes of an object of type
     type-name.
```

There are also quite a few binary operators available for C:

```
x*y - Multiply.
      The result is the product of x and y.

x/y - Divide.
      The result is the quotient of x divided by y.

x%y - Remainder.
      The result is the remainder of x/y.

x+y - Add.
      The result is the sum of x and y.

x-y - Subtract.
      The result is the difference of y from x.

x<<y - Shift left.
       The result is x shifted left y bits.

x>>y - Shift right.
       The result is x shifted right y bits.

x<y - Less than comparison.
      The result is 1 if true, 0 otherwise.

x>y - Greater than comparison.
      The result is 1 if true, 0 otherwise.

x<=y - Less than or equal comparison.
       The result is 1 if true, 0 otherwise.

x>=y - Greater than or equal comparison.
       The result is 1 if true, 0 otherwise.

x==y - Equal to comparison.
       The result is 1 if true, 0 otherwise.

x!=y - Not equal to comparison.
       The result is 1 if true, 0 otherwise.

x&y - And.
      The result is the bit-wise and of x and y.

x y - Exclusive or.
      The result is the bit-wise exclusive or of x
      and y. Be careful not to confuse this with the
      "power of" facility of Basic.

x y - Inclusive or.
      The result is the bit-wise inclusive or of x
      and y.

x&&y - Logical connective and.
       The resuslt is 0 if x is zero, without
       evaluating y. Otherwise the result is 1 only
       if both x and y are non-zero. This is used in
       logical expressions such as:

  if (a==b && b==c) a=c;

Evaluation stops as soon as the expression
       becomes false.

x  y - Logical connective or.
       The result is 1 if either x or y is non-zero.
       This is used in expressions such as:

       while (a!=b    b!=-c    d!=q) ++a;

t?/x:y - Conditional evaluation.
         This acts like the statement:

          if (t)
                (
                x;
                )
          else
                (
                y;
                )

The result is the result of the evaluation.

x=y - Assignment.
      The result (which is x) is the value of y.

x*=y - Equivalent to x = x * y.

x/=y - Equivalent to x = x / y.

x%=y - Equivalent to x = x % y.

x+=y - Equivalent to x = x + y.

x-=y - Equivalent to x = x - y.

x<<=y - Equivalent to x = x << y.

x>>=y - Equivalent to x = x >> y.

x&=y - Equivalent to x = x & y.

x =y - Equivalent to x = x   y.


x =y - Equivalent to x = x  y.

x,y  - x is evaluated first, then y.
       The result is the value of y after evaluation.
```

Although these operators make writing programs much easier and less verbose, their conciseness tends to make the programs difficult to read.

Variable and function names can generally be of any length, and use any displayable characters, as long as they do not conflict with the reserved words. Data in C is organized in both types and classes. A storage class is a method of storing the data. C has the following storage classes:

extern — Specifies that an external definition for the given identifier outside of that file.

auto — Specifies that the given identifier will exist as dynamic local variable. It will be created when that block of code is executed, and destroyed when the block is exited.

static — Specifies that the given identifier will not be known outside a block, but will remain inside that block. It differs from auto in that the data declared static will always exist until the termination of the program. This is similar to Algol's "own" declaration.

register — Means the same as auto, but specifies that efficient storage, such as registers be favored to hold the object, and the address of the object (&x) cannot be taken.

typedef — Specifies that the identifier should be recognized as a type specifier. This is used to create new data types from old ones. This is similar to Pascal's "type" declaration.

Data types are relatively primitive in C. Arrays are constructed of scalars; many implementations do not have them at all. The following is a list of data types that might be available:

char — A byte integer used to hold a single character of the machine's character set.

NEW LOWER RAM PRICES

LDP1/2

The LDP1/2 utilizes the advanced 8088 processor to provide up to 8 times the throughput of a 4 MHz Z80A processor. The powerful instruction set of the 8088 is ideally suited to higher level languages such as PASCAL and PL/1. The 10 slot motherboard leaves 7 slots for USER expansion. With the option of a 10 MByte Winchester and MP/M-86, the LDP mainframe becomes a powerful multiuser system with the capability of handling 8 users without the degradation in performance experienced with Z80 CP/M systems. The performance of the LDP1 and LDP2 has never been available for such an affordable price.

FEATURES:
— LDP88, 8088 CPU board
— LDP72, advanced floppy disk controller
— LDP128K dynamic RAM
— 1 serial RS232 port
— 10 slot mainframe
— 18" floppy drive (LDP1), 2 8" floppy drive (LDP2)
— 4K EPROM socket for user population
— ROM resident diagnostics
— CP/M-86

OPTIONS:
— HAZITALL
— 8" 10 MByte Winchester
— MP/M-86 multiuser system
— Woodgrained 7 slot chasis

PRICES:

LDP/1 w/128K RAM	$3295
LDP/2 w/128K RAM	3895
LDP/2 Winchester, 128K RAM	6795

SOFTWARE:

CP/M-86	$250
BASIC	450
FORTRAN	450
FORTH	100
C	150

LDP88 8088 CPU BOARD

• 8088 CPU 5 MHz operation upgradeable to 8 MHz • 9 vectored interrupts • Fully complies with IEEE 696 electrical and timing specs • RS232 serial port with modem controls • 1K bytes of static RAM • 2 EPROM sockets (2716 or 2732) • 8087 upgrade kit available in Sept. • 8 bit bus eases interface to other S100 bus boards • 1MByte address space • 65K I/O ports

PRICES: Kit $319.95
Assembled & tested $349.95

64/256K MEMORY

• 8 or 16 bit operation • Meets all IEEE 696 specs • Access time 350 ns from PSYNC low • Intel 8203 dynamic RAM controller • 24 or 16 bit address decoding • No wait states with 5 MHz 8088 or 8086 • Parity with Error interrupt generation • No DMA RESTRICTIONS • 64K board is upgradeable to 256KByte board

PRICES: 64K $595
128K 795
256K 1395

HAZITALL

• 2 Serial RS232 ports • 2 parallel ports with handshake control • Math processor support (8231/9511 or 8232/9512) • WINCHESTER DISK support • Real time programmable interrupt • Clock/calendar with battery back up • Synchronous data communication supported

PRICES: Kit $249.95
Assembled & tested $325.00

LDP72 FLOPPY DISK CONTROLLER

• IBM compatible single and double density format • Single or double sided drives • Programmable data record length (128 to 8192 bytes/sector) • Multi sector and multi track transfer capability • Parallel seeks on up to 4 drives • On board digital data separator • Software selectable single or double density operation • Separate connectors for 5 1/4" and 8" drives • Software selection of standard or minidrives allowing mixing of both drives on a single controller

PRICES: Kit $219.95
Assembled & tested $274.95

Watch this ad for the announcement of the most powerful S-100 Bus CPU board available.

Lomas Data Products
11 Cross Street
Westborough, MA 01581
Telephone (617) 366-4335

int — Usually a two byte integer, can also be called "short" in some versions.

long — Usually a four byte integer.

float — A floating point number, usually four bytes.

double — A double precision floating point number usually eight bytes.

struct — A sequence of one or more types used to create logical records in a program. Similar to Pascal's "record" and COBOL's record. In some implementations can be used to access individual bits in a byte as a field.

union — A record specified by fields.

pointer to — An unsigned integer used to hold an address of some object.

```
char line[80], *pline;
```

*pline is the pointer to declaration.

array of — A repetition of some type. Example is above in line 80 declares 80 repetitions of type char, and aligned it on line.

Pointers provide a very powerful and sometimes confusing tool in program writing. Simply stated, a pointer points to some location in the machine. They can always point to any location in the machine's address space. A pointer can be manipulated, or by placing an asterisk in front of the name, that which is being pointed to can be manipulated. For example:

```
char fcb[12], *pfcb;
```

Declares an array of 12 bytes with the first byte called fcb. If fcb is located at 54A2h, then the following would set the pointer pfcb to 54A2h:

```
pfcb=&fcb;
```

Note the value of pfcb has been changed to 54A2h. If we wanted to increment the pointer to point to the second byte, the following could then be done:

```
++pfcb;
```

Now, if we want to set what is now at 54A3h to "?", then the following could be done using indirection through pointers:

```
*pfcb='?';
```

This is where C can be confusing:

```
*++pfcb='?';
```

does the same thing. It increments pfcb to 54A3h and puts a "?" at that address.

C is not perfect however, there are some major disadvantages as well. One is the problem of operators being intermixed in confusing ways, as with *++pfcb.

Another rather serious problem is in the definition of the language itself. There is no actual formal definition of the C programming language. The closest is in a book written by the designers of the language [Kernighan 1978] in which there are quite a few vague points where some things are not absolutely explicit. Because of this, some implementations may differ in points of interpretation.

The next problem is that of portability. Although many implementations claim to be transportable to systems using compilers other than their own implementations, such as UNIX systems, they may in actuality be transportable to only a few other systems. This is not because of the syntax of the language, which will probably be identical, but because of the definition of the functions used. It may be necessary to rewrite some of the functions used to do what you expect them to do.

Lastly, and this is a mixed blessing, the C compiler is not overly picky on the whole. It will allow types to be confused without doing anything about it, and this can lead to difficult code to read, as well as results that may be CPU dependent. On the other hand, if you want to do something fancy on a specific machine, it will allow it.

Although there are drawbacks, C is a very powerful higher level language that is simple to use. It is one of the best for doing any form of systems software, and has quite a large user following.

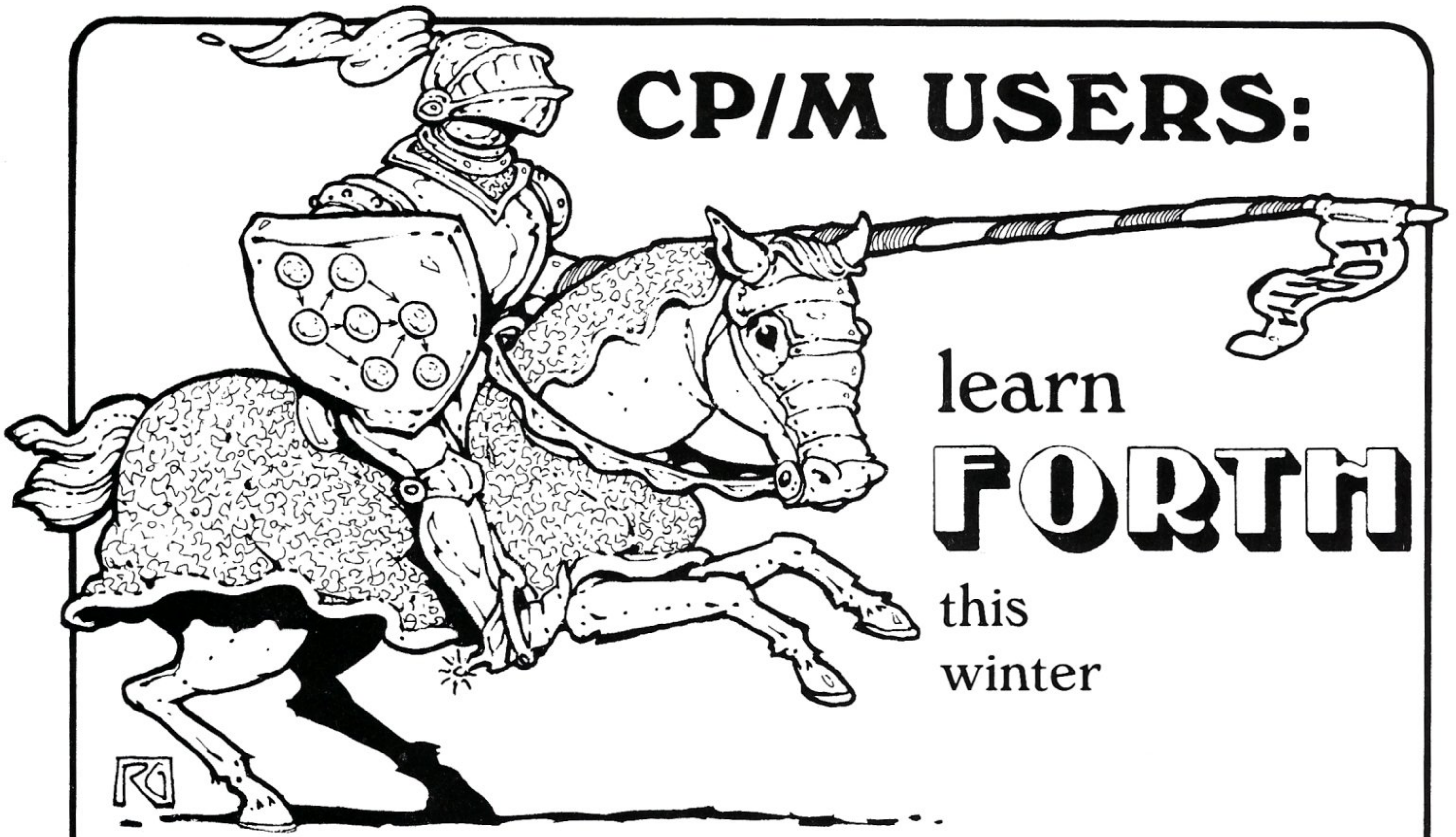
Name: BD Software C Compiler
Price: $145
Distributed by:
Lifeboat Associates
1651 Third Avenue
New York, NY 10028

The BDS C Compiler was written by Leor Zolman of Cambridge, MA, and is distributed by Lifeboat Associates in New York City. This compiler takes the large step from a small development language to a full scale production compiler. Although not nearly as large or complex as compilers residing on mainframe computers, the BDS C compiler can handle a significant portion of the programming requirements of a microprocessor-based small computer.

Unlike Basic, it is not an interpreter but an actual compiler that generates machine code. Unlike most Pascals, it is a native code compiler. That is, it generates machine code directly executable on the machine for which it was designed (in this case the 8080, Z-80, and 8085 microprocessors) without generating an intermediate source code such as assembler source code or Pascal "P-code".

The BDS C package comes with a CP/M-compatible floppy disk containing a version of the compiler (in this case version 1.43) and the linker. In addition, all of the support modules containing compiled or assembled functions, as well as the run-time package, are also included. Although the compiler itself is not furnished in source form, the C run-time package and the two standard function libraries written in C, and the standard assembly language function libraries are furnished in ready-to-run versions and as source files. Since any C program is (or should be) made up of many separate functions, the BDS C package contains a library manager called CLIB that allows the programmer to build library files out of compiled functions from many files.

Although the BDS C Compiler does not directly support floating point numbers, a set of functions is included that allow floating point numbers to be manipulated in a reasonable manner when necessary. Another set of functions allows the programmer to integrate assembler code into callable functions. These routines require the assistance of the Digital Research Macro Assembler however, which is not included on the disk. Lastly, a



CP/M USERS:

learn **FORTH** this winter

Why learn FORTH? To do **BIG** jobs in little time!

— Timin FORTH release 2, a complete FORTH system (includes user manual/tutorial, resident editor & assembler)	$95
— Manual only, credits toward diskette purchase	$20
— Disk of FORTH application modules, by Scott Pickett (Full disk of FORTH source code. Clear, detailed documentation)	$75
— Visual Image Editor, supplied in FORTH source code	$75
— CP/M Utility Package, supplied in FORTH source code	$50
— Timin FORTH release 3, a complete FORTH system (many new features, plus Visual Editor & CP/M package above) (Release 3.5, same price, has FORTH screens in CP/M file.)	$235
— FLOATING POINT MATH option for release 2, 3, or 3.5	$100
— ROMable FORTH, for engineers producing ROM code	$350
— BOOKLET explaining FORTH, mailed same day, send cash	$2

Above are 8" prices. Add $15 for 5" North Star, Heath, Microp, etc.

Timin Engineering Company
9575 Genesee Avenue, office E-2
San Diego, CA 92121
Orders: (714) 455-9008
Technical Hotline: (714) 455-9041

WE HONOR master charge THE INTERBANK CARD

VISA

number of example programs are included, such as a working Othello and a rather nice terminal emulation program called "Telnet." Once Telnet has been customized for a particular computer, it provides intelligent terminal operations such as file transfer and receive. With another computer also running Telnet, it will transfer compiled files as well. I have run Telnet for quite some time to provide communication between my computer's acoustic coupler and a PDP-2060, as well as a PDP-1050, and Micronet.

In addition to the floppy disk software and documentation, BDS C comes with a seventy page manual that describes the BDS implementation, how it varies from previous BDS C versions, how to use the compiler, the linker, and the librarian. Also listed is usage documentation of the user available functions. Finally, comparision of BDS C to the "standard" C specified in Kernighan and Ritchie is also included.

Included in the BDS C package is a book by the authors of the original C language from Bell Labs, *The C Programming Language*, by Brian W. Kernighan and Dennis M. Ritchie. If anyone seriously intends to program the authoritative reference on C and is a necessary item in any C programmer's library.

In addition to the floppy disk software and documentation, BDS C comes with a seventy page manual that describes the BDS implementation, how it varies from previous BDS C versions, how to use the compiler, the linker, and the librarian.

BDS C complies with the syntax specified in Kernighan and Ritchie and supports many of the features of C—with some exceptions. These include short int, long int, float and double. Also there are no explicitly declared storage classes. Static and register classes do not exist; all others are either external or automatic. That is, if a function is called and it is not in that file, it is simply left as a reference to be resolved by the linker. Initializers are not allowed, so you cannot declare a variable and assign it a value in the declaration section. Unary operators "(type-name) expression" and "sizeof (type-name)" are not implemented. In structure and union declarations, bit fields are not implemented.

However, the preprocessor functions #define, #include, #ifdef, #ifndef, #else, #endif, and #undef are implemented.

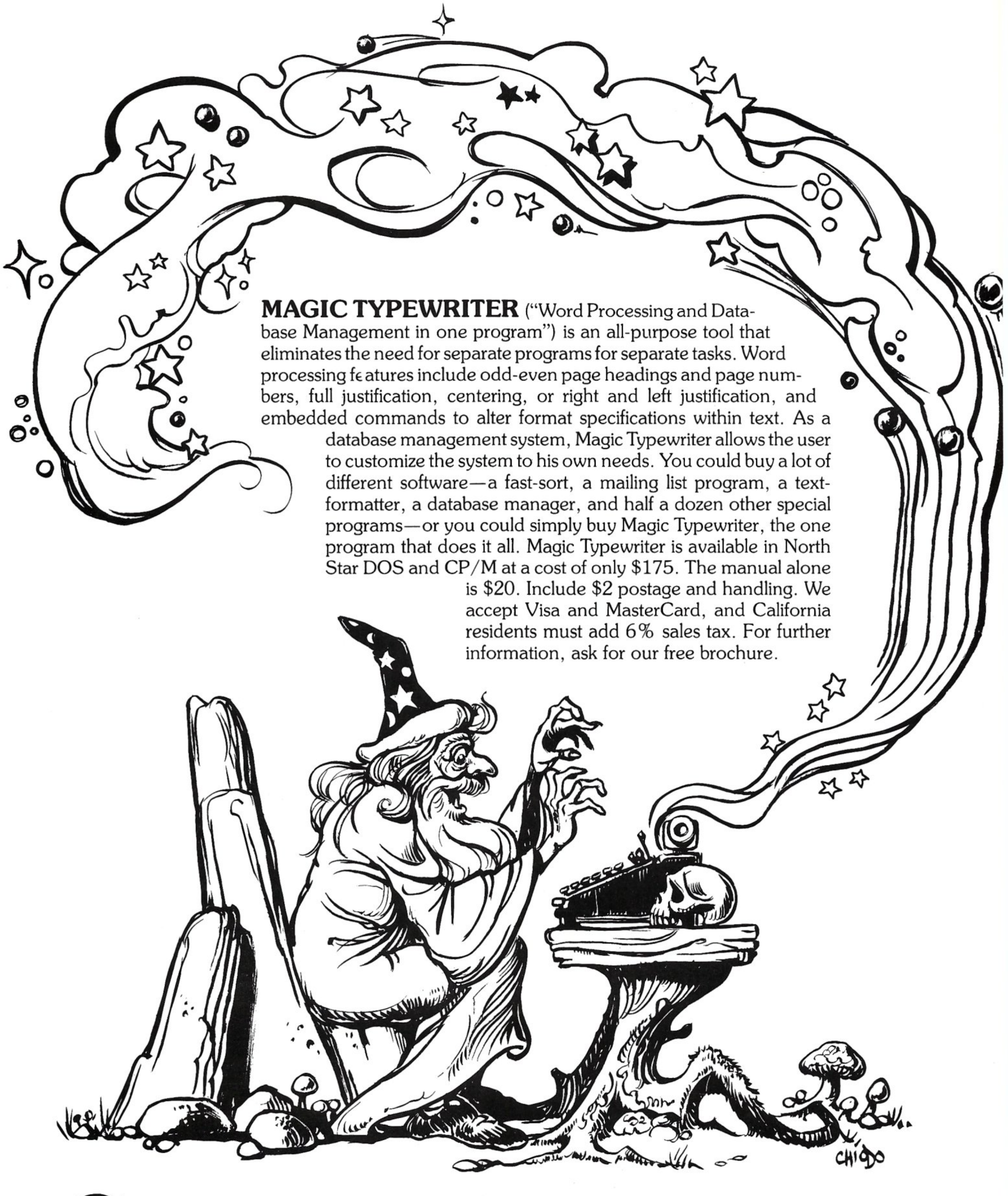
The functions available with BDS C are reasonably complete with many functions from standard C, such as printf, getc, ungetc, scanf, and fopen. Simple machine-oriented functions that call the CP/M BDOS and the BIOS are available. Peek and poke memory locations, input and output from ports, call assembly language functions, seed and get random numbers, fill a block of memory, move a block of memory, sort a set of elements, execute a CP/M program, and a primitive set of storage allocations functions are also included. Character input and output, with and without formatting is available. Many string and character processing functions are included, as well as file input/output functions. Lastly, a set of plotting functions for DMA video boards such as the Processor Tech VDM-1 are included (I have not tried them).

BDS C compiles in three stages. The first stage is the parser and the second is the code generator. The output of the code generator is a C Relocatable (CRL) file. This CRL file then has to be linked with all of the functions and such to generate the final runable file using CLINK. If no options are specified, CLINK produces a COM file that is executable by CP/M. There are options available in both the compiler and linker that allow programs to be executable at addresses other than 100 Hexadecimal as CP/M requires. This is a great advantage to those people who must write software for other systems or system software for CP/M. The BDS C compiler compiles in one chunk, loading the entire source file into memory at once as opposed to loading the source in segments, working on it a bit, and producing the CRL file. That means that the maximum size of a source file is limited by the size of main memory. This problem can be circumvented by writing programs with a lot of functions and then compiling the functions separately. I have done quite a bit of work with large C programs using many functions, and have had no problem using the compiler in 48K bytes of available memory.

The compiler is simple to use, and compilation and linkage is fast. I have written programs in BDS C and assembler of equal function and found the BDS C compiler to produce executable code in less time than CP/M's assembler. Unfortunately, the assembler produces smaller code than the compiler, since every BDS C program must include approximately 2K of run-time package with every program.

When I first tried to bring up version 1.32 of the compiler, I had no problem. I then received version 1.43 from Leor Zolman, and was unable to link any of the files that I had compiled. I called Leor up and told him of the problem. Since I had to go into Boston anyway, I decided to bring in my non-working version and pick up a working version. I stopped at his place where he gave me a disk that was supposed to be working. I went home to try it out; and when I did, it still didn't work (it proceeded to happily crash my system every time I tried). I called Leor and explained all of the symptoms. He had no idea what was wrong, but offered to make the forty plus mile trip to Worcester (which is horrible in itself) by motorcycle on a very cold day to see if he could figure out what was wrong. To make a long story shorter (I hope), he came out, we tried to figure out what was wrong, and it turned out I had a bad copy of the run-time package.

Other than that one incident, I have been impressed with the operation of the compiler. An interesting note about the BDS implementation is its user following. I have met many people who have been highly impressed by the compiler, and there is also a user group for it. In addition, both the Small C manual and Tom Gibson of Tiny C recommend the BDS C compiler as well. It is a complete and inexpensive implementation of a C compiler



MAGIC TYPEWRITER ("Word Processing and Database Management in one program") is an all-purpose tool that eliminates the need for separate programs for separate tasks. Word processing fe atures include odd-even page headings and page numbers, full justification, centering, or right and left justification, and embedded commands to alter format specifications within text. As a database management system, Magic Typewriter allows the user to customize the system to his own needs. You could buy a lot of different software—a fast-sort, a mailing list program, a text-formatter, a database manager, and half a dozen other special programs—or you could simply buy Magic Typewriter, the one program that does it all. Magic Typewriter is available in North Star DOS and CP/M at a cost of only $175. The manual alone is $20. Include $2 postage and handling. We accept Visa and MasterCard, and California residents must add 6% sales tax. For further information, ask for our free brochure.

CDE CALIFORNIA DIGITAL ENGINEERING
P.O. BOX 526 ★ HOLLYWOOD, CA 90028

and I would recommend it to anyone intending to do any serious work in the C programming language.

Name: Small C Compiler
Price: $15
Distributed By:
The Code Works
P.O. Box 550
Goleta, CA 93017

The Small C compiler first made its debut in an article published by Ron Cain in the May 1980 issue *Dr. Dobb's Journal*. He opened the article by saying that he had to have a compiler for his home computer. Since Cain did not have CP/M and did not wish to spend the money for the expensive compilers that were on the market, he decided to write his own. He chose to write his 8080 flavored C compiler in C, first using the Tiny c interpreter, then using a full UNIX C compiler. While writing it, he was careful to be sure that all of the code used in writing the compiler would potentially be able to be compiled by that compiler, which would allow it to be modified once installed in the target system. Upon completion of that project, he decided to publish the entire compiler, written in C, in *Dr. Dobbs Journal* along with a discussion of the operation of the compiler. For someone who really needed a compiler, this was great—that is if you had access to a machine with a C compiler on it and if you wanted to type in ten pages of (two columns each) of very, very small print.

At about this time, a company called The Code Works from Goleta, CA entered the scene. They took Ron Cain's Small-C compiler and interfaced it to CP/M. They added to it an unpublished (at that time) run-time system, a few sample programs, and an eleven page instruction manual. They put it all, minus the manual, on a standard CP/M eight inch floppy disk, which included all of the source code for everything, and a working compiled version of the compiler that generated code acceptable to CP/M's assembler. After all that, they started to market it, selling it for an unprecedented $15 plus two dollars postage.

When their press release came to *Microsystems* for publication, I thought something had to be in error. I hadn't seen any compiler software commercially sold for CP/M use at anywhere near that price. When I called them, I was assured that this was in fact the price of the package and not a typographical error.

Once I examined the Small-C package, I was impressed. There was none of the quick, throw-it-together programming and documentation I have found in some of the other software available at that price range. Instead I found a clear, concise manual describing what you could and could not do with this compiler, including a warning that it was not suited for major systems programming tasks. It explained the way its functions were used, and how to interface to assembly language. At the end, the manual gave a specification of what features were available from standard C.

Small C does in fact comply with the "standard" C syntax defined by Kernighan and Ritchie, however its function calls do not comply with standard nor does it support many of the nicer features of C. It has no storage class specifications. The compiler is limited to two byte integer and single byte character types, as well as pointers. In addition, single dimension vector arrays are available. Most unary operators are supported, with the exception of not "!", complement "^ ", and "sizeof". Most binary operators are supported as well, with the exception of connectives and "&&" and or " || ". This means that such statements as:

```
if (a<b && c>d) ++b;
```

cannot be used. The if-else abbreviation "?." is also not available, but is not really a great loss. The concatenated assignment operators "+=", "-=", "*=", "/=", "%=", " >> ", " << =", "&=", "^ =", and "| =" are not available. This implementation allows very limited structured flow using the "if" and "while" statements. None of the other convenient statements "for," "do while," "switch," "case," "default," and "goto" have been included.

In a small program, many of the above features are not actually necessary, but the lack of ability to logically connect conditions, coupled with a lack of the "for" and "switch-case" operations can make larger programs cumbersome and difficult to program.

As for the I/O functions available, they are rather limited, but usable. Available functions allow single characters and character strings to be read from and written to the user's console. Disk files can be opened and closed, and single byte data can be read in from, or written to any of four open files. Disk I/O is only buffered in 128 byte sectors, however. Another function allows the user to perform CP/M BDOS calls from C.

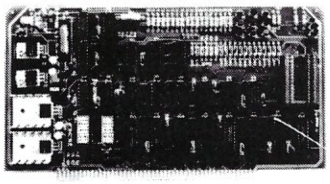
In addition to the above functions, there is a large amount of generally useful functions inside the compiler source file. It is a shame that these were not pruned from the compiler source and distributed with the other user functions, since they already exist. One such function, called match, compares a specified string with the input stream, and returns true if the two match. This was used in the lexical analysis phase of the compiler, but could just as easily be used in other application programs.

There is a nice feature available that allows inline assembly code to appear in the C source code. Since the compiler generates CP/M 8080 assembler code as its output, inline code can be easily added to the compiler. Any time an "#asm" is encountered in the source, the compiler just takes input and passes it to output with no processing. Once an "#endasm" is encountered, the compiler continues compiling. When I first received the compiler, I had some problems with the assembly code linkages in areas where the manual was unclear, but after a call to The Code Works, I was able to understand it. After my call, they wrote a sample program detailing that linkage, and included it on the distribution disk, so there should be no further problems with other customers.

The Small-C compiler is a one pass compiler (though, technically, the assembler could be considered a second pass). This means that there isn't any checking of defined symbols—everything is just passed on for the assembler to catch. In addition, there is no optimization done on compiled source, either globally or locally. This tends to generate rather inefficient object code, taking more space in memory, and more time to run. In small applications,

PUT YOUR SYSTEM ON THE FOREFRONT OF TECHNOLOGY

PROM BLASTER

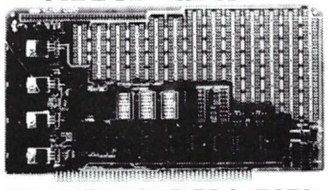


PROGRAMS MOST FAMILIES OF EPROMS!

- Accepts 1K/2K/4K or 8K EPROMS
- Extended Device Option
- Phantom Slave Option
- All Programming Characteristics Software Controlled
- Accepts Single or 3 Supply Parts
- Device Address Switch Selectable

Kit Price $199.95

KLUGE CARD

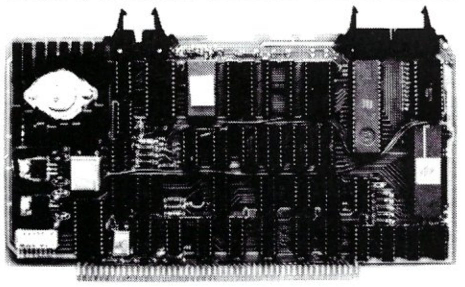


SIMPLIFY YOUR PROJECTS WITH A PROTOTYPE BREADBOARD WITH EXTRAS

- 4 On-Board Pwr. Supplies up to 3 of which can be +5, or ±5, ±12
- Switch Selectable Memory or Device Address
- On-Board Address/Device Decoding
- Bi-Directional Data Bus Buffering
- On-Board Wait States

Bare Board $39.95

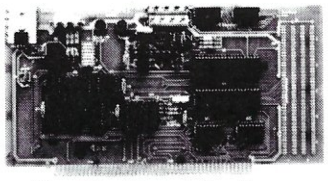
6809 SINGLE BOARD COMPUTER



- 2K ROM
- 4K/8K/16K RAM
- 20 Parallel I/O Lines
- RS 232 Interface
- Baud Rates from 110 to 9600
- All Memory/I.O. Relocatable on 4K Boundaries
- 8080 I/O Instructions Memory Mapped (Gives 256 I/O Ports)
- Complete Documentation
- ADSMON Monitor Includes User Callable Functions, Autopatch and more

Kit Price $299.95 (Includes Software)

NOISEMAKER



- Two On-Board Audio Amplifiers for Stereo Sound Effects
- Uses the GIAY 3-8910
- Six Tone Generators
- Two Envelope Generators
- Two Noise Sources
- Four 8-Bit I/O Ports
- Up to Two Wait States

ALL SOUND EFFECTS ARE SOFTWARE CONTROLLED FOR AN ENDLESS VARIETY!
KIT $84.95

MOTHER BOARDS

THREE S-100 MOTHER BOARDS AVAILABLE;

- 3 Slots for Stand-Alone Applications
- 9 Slots for Small Systems
- 18 Slots for Full Size Systems
- Active Termination (9 & 18 Slot Boards Only)
- Extensive Ground Shield
- Breadboard Area

(Write for complete information)

The products shown here are just an example of the quality products and service **Ackerman Digital Systems** offers. **ADS** is an engineering company dedicated to providing the micro computer industry with solid quality products to meet a variety of needs.

All **ADS** products adhere to the **I.E.E.E. 696/S-100 Bus** standard, a bus noted for its widespread acceptance and versatility.

For catalog containing complete hardware and software information, write Ackerman Digital Systems, Inc., 110 N. York Rd., Elmhurst, IL 60126. TEL 312-530-8992

ads
ACKERMAN DIGITAL SYSTEMS, INC.

this is relatively unimportant, but as the programs increase in size, it can be a real problem.

I found the compiler easy to run, and with no apparent bugs. Unfortunately, compilation is rather sluggish. The run-time package that is included is written in assembler with #asm and #endasm at each end. For a program to assemble, this file must be appended onto the assembler source output of the compiler. The method suggested in the manual is to simply run it through the compiler along with the program being compiled. Doing that is a rather time-consuming process, and I found it much easier to remove the #asm and #endasm statements, and once the compilation was finished, append the run-time software to the end of the compiler output with a text editor. The disadvantage of this is that the entire run-time package is appended, taking up more space in memory.

I was able to compile the compiler source in fourteen minutes on a 4Mhz Z80-A. It took another eight and one half minutes to assemble and 45 seconds to load. After going through all of that, it did run. The source code of the actual compiler was reasonably well written and should be easy to modify. It is also a good example of what a simple, "bare-bones" compiler looks like. In looking at the source, it does not seem that any special parsing or lexical analysis method was used, creating a few additional unnecessary states in code generation.

For anyone looking for a simple compiler to do smaller programming tasks, or for a minimum expense language, the Small-C compiler is a very good choice at $15. For an additional programming tool a few steps above assembler, it is also a good choice. If larger applications are expected, I would not choose this, but go with one of the larger compilers costing from about $145 to $700.

Name: tiny c TWO Compiler
Price: $250
Distributed by:
tiny c associates
P.O. Box 269
Holmdel, NJ 07733

Tiny-C Two Compiler

Tiny-c has been well known for its interpreter of a dialect of the C language, and for its excellent documentation for beginners. After the interpreter was on the market for quite some time, the folks at tiny-c associates felt that although the original tiny-c had quite a following, it was relatively slow for large projects (as an interpreter would be). They decided that a compiler would be very useful.

Now, think about the interpreter compiler combination for a given language. An interpreter is nice and easy to use, and programs can be written, tested, debugged, retested, and so on in the single environment of an interpreter. But interpreters are slow. On the other hand, compilers are faster by at least one order of magnitude, but they are not as simple to work with. You have to enter an editor, type in the program, exit the editor, and attempt to compile. If it won't compile due to syntax errors, you have to reload the editor and edit the program again. When you manage to compile your program, you have to load the linker and debug all of the logic errors. What would offer the best of both worlds would be a good interpreter to make program development easier, and once it works, a compiler to make the program run faster. For those of you who program in Basic, one combination like this is the Microsoft Basic-80 interpreter, which makes development easier. When you want it to cook right along you can use the Microsoft Basic compiler. However, Basic is not the best language for all things, and it would be convenient to have a somewhat structured language do the same thing. The tiny-c 1 interpreter, combined with tiny-c TWO (the compiler), make a strong step in that direction—although the compiler comes with some very powerful functions not supplied in the interpreter.

Tiny-c is a language that is similar to the C programming language. It is important to realize, however, that tiny-c is a different language from C. It has many constructs and data types to those similar in C, but the syntax and operation are slightly different. Where a program in C is written as:

```
main()
{
        some set of operations
}
```

a tiny-c program is written as:

```
main
[
        some set of operations
]
```

Comments in C are written with a beginning /* and end with a */, such as:

```
getch()
/*
        This routine returns a character from the
        console
 */
 {
        some operations
 }
```

while comments in tiny-c start with /*, and end with the end of the line, such as:

```
getch
/*
/*      This routine returns a character from the
        console
/*
[
        some operations
]
```

In the C language, statements end with a semicolon ';':

```
range=big-little+1;
```

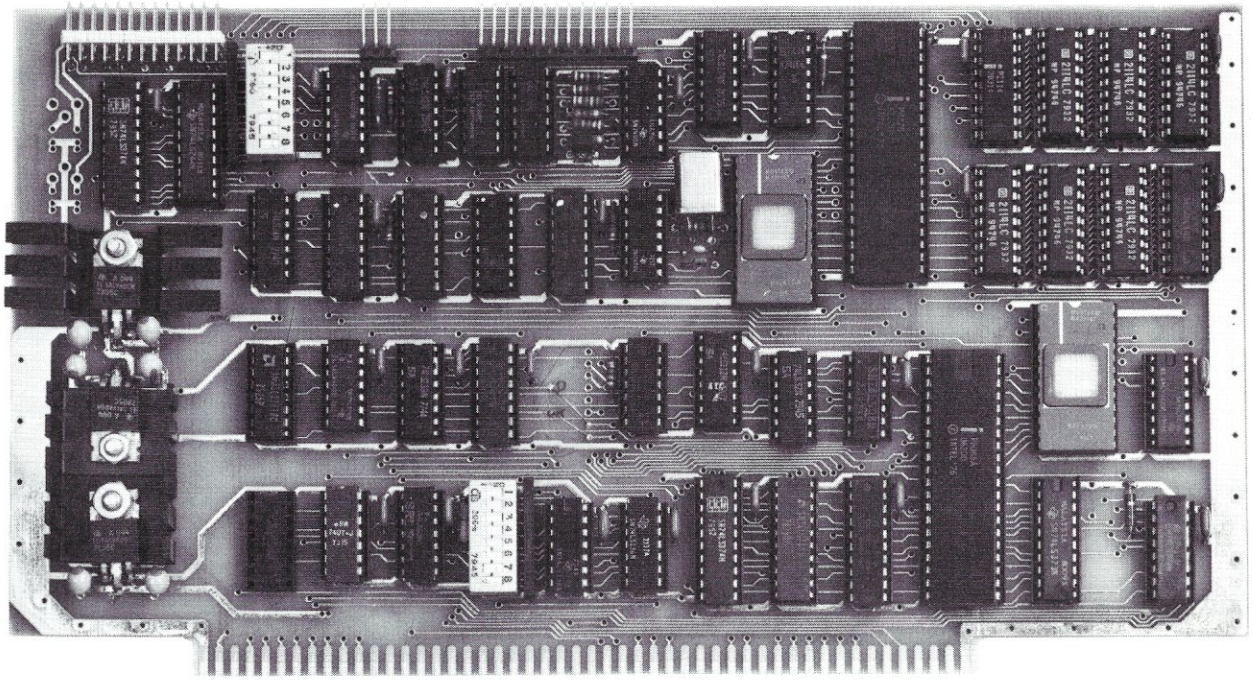
where in tiny-c, the end of a line or a semicolon signifies the end of a statement, although more than one statement may be on a single line is separated by a semicolon:

```
range = big-little+1
last = last*seed

     -or-

range = big-little+1 ; last = last*seed
```

In C, all compound statements are delimited by the curly braces { and } :

```
{
        x=x-1;
        a=b+c;
        b=b*2-a;
        x=b-a;
}
```

INTELLIGENT VIDEO I/O FOR S-100 BUS



VIO-X

The VIO-X I/O Interface for the S-100 bus provides features equal to most intelligent terminals both efficiently and economically. It allows the use of standard keyboards and CRT monitors in conjunction with existing hardware and software. It will operate with no additional overhead in S-100 systems regardless of processor or system speed.

Through the use of the Intel 8275 CRT controller with an onboard 8085 processor and 4k memory, the VIO-X interface operates independently of the host system and communicates via two ports. The screen display rate is effectively 80,000 baud.

The VIO-X1 provides an 80 character by 24 line format using a 7 × 9 dot matrix to display the full upper and lower case ASCII alphanumeric 96 printable character set (including true descenders) with special characters for escape and control characters. An optional 2732 character generator is available which allows an alternate 7 × 9 contiguous graphics character set.

The VIO-X2 offers an 80 character by 25 line format using a 9 × 9 dot matrix allowing high-resolution characters to be used. This model also includes expanded firmware for block mode editing.

Both models support a full set of control characters and escape sequences, including controls for video attributes, cursor location and positioning, cursor toggle, light pen location, and scroll speed.

Video attributes provided by the 8275 in the VIO-X include:

- FLASH CHARACTER
- INVERSE CHARACTER
- UNDERLINE CHARACTER or
- ALTERNATE CHARACTER SET
- DIM CHARACTER

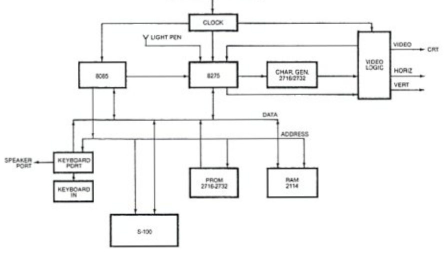
The above functions may be toggled together or separately.

The board may be addressed at any port pair in the S-100 host system. Status and data ports may be swapped if necessary. Inputs are provided for parallel keyboard and for light pen as well as an output for audio signalling. The interrupt structure is completely compatible with Digital Research's MP/M

FEATURES

- HIGH SPEED OPERATION
- PORT MAPPED S-100 INTERFACE
- FORWARD/REVERSE SCROLL or
- PROTECTED SCREEN FIELDS
- CONVERSATIONAL or BLOCK MODE
- INTERRUPT OPERATION
- CUSTOM CHARACTER SET
- CONTROL CHARACTERS
- ESCAPE CHARACTER COMMANDS
- INTELLIGENT TERMINAL EMULATION
- TWO PAGE SCREEN MEMORY

VIO-X1	80 × 24	7 × 9	A & T	**$295.00**
Conversational & Limited Block Modes				
VIO-X2	80 × 25	9 × 9	A & T	**$345.00**
Conversational & Block Modes				



VIO-X S-100 I/O INTERFACE

FULCRUM™ COMPUTER PRODUCTS

Distributed by:
WW COMPONENT SUPPLY INC. 1771 JUNCTION AVENUE • SAN JOSE, CA 95112 • (408) 295-7171

In tiny-c they are delimited by square braces [and]:

```
[
        x = x-1
        a = b+c
        b = b*2-a
        x = b-a
]
```

Both allow many levels of nested compound statements.

Functions in C are specified by the function name, a left parenthesis, a list of argument names, and a right parenthesis. This is followed by a type description of each of the arguments:

```
nonprint(c)
char c;
/*
        This function will return true if the character
        is non printable.  It only checks the low order
        seven bits.
*/
{
        return c&128<=32 || c==128;
}
```

It might be called by the statement:

```
if (nonprint(byte))
{
        some statements
}
```

In tiny-c, a function is declared by the name of the function, followed by a type description of each of the arguments.

```
nonprint
char c
/*
/*      This function returns true if the character
/*      is non printable.  It only checks the low order
/*      seven bits.
/*
[
        return c&128<=32
]
```

The function can be called as:

```
if (nonprint(byte))
[
        some statements
]
```

If a function in C returns has no arguments, such as getch, it has to be called like:

```
c=getch();
```

with both parentheses included. In tiny-c, it could be written as:

```
c=getch
```

without the parentheses.

The last major syntactic difference is in the usage of subscripts in arrays. In C, an array is specified by:

```
a=num[index];
```

while in tiny-c:

```
a=num(index)
```

substituting the parenthesis for the square brackets.

For comparison, the following is a small program written in tiny-c and C, taken from the *Tiny-c Newsletter*, Number 1, February, 1981:

```
tiny-c                                  C
------                                  -----
main                                    main()
char a(o)                               char a[o];
[                                       {
        int k                                   int k;
        while (++k < 10)                        while (++k < 10)
        [                                       {
                pl""                                    pl("");
                pn k                                    pn(k);
                ps ""                                   ps("");
                putchar a(k)                            putchar (a[k]);
        ]                                       }
]                                       }
```

The purpose of that newsletter was to explain to users how to convert the tiny-c TWO compiler to a compiler that would compile code closer to standard C. Since the compiler comes with all of the source code, the user only has to make the necessary changes in the code and recompile the compiler. For those who don't wish to make changes themselves, tiny-c associates will be distributing a modified version containing the changes, which they call C-TWO. One of the problems of tiny-c to the experienced C programmer is getting used to and converting old programs to the tiny-c syntax. The C-TWO modification could greatly ease the relearning. For the novice programmer, both to C and to structured languages as a whole, tiny-c is much easier to learn—first with the interpreter, and then with the compiler.

Now that we have looked at the syntactic differences between C and tiny-c, we will look at the functional differences. This is a comparison between the standard C [Kernighan 1978] and tiny-c. Although tiny-c supports many of the features of C, some important items are missing—such as the convenient switch/case statement. In addition, the do/while statement is missing, although the while statement has been implemented. Also missing is the for statement, but this can easily be done in a while loop. A for statement might look like:

```
for( i=1 ; i!=10 ; ++i)
{
        do something
}
```

which can be implemented with a while statement in tiny-c as:

```
i=1
while (i!=10)
[
        do something
        ++i
]
```

It's not as pretty, but it works.

A switch/case statement, on the other hand, is not as easily implemented, nor does it look as nice:

```
switch(num)
{
        case 1:
                        statement_1;
                        break;              /* exit switch */
        case 2:         statement _2;
                        break;
        case 3:         statement _3;
                        break;
}
```

Z-80 and 8086 FORTH

Z-80 FORTH — a complete program development system. Uses standard CP/M® compatible random access disk files for screen storage. Package includes: interpreter/compiler with virtual memory management, line editor, screen editor, Z-80 Assembler, decompiler, utilities, demonstration programs, and 80 page user manual. System requirements: Z-80 microcomputer, 48 kbytes RAM, CP/M 2.2 or MP/M 1.1....... **$50.00**
With software floating point arithmetic **$150.00**
With AMD 9511 support routines **$150.00**

Z-80 FORTH WITH NAUTILUS SYSTEMS CROSS-COMPILER. Extend/modify the FORTH runtime system, recompile on a host computer for a different target computer, generate headerless code, generate ROMable code with initialized variables. Supports forward referencing to any word or label. Produces load map and list of unresolved symbols. 107 page manual. System requirements as for Z-80 FORTH above ... **$200.00**

8086 FORTH with line editor, screen editor, assembler, and utilities. Uses standard CP/M compatible random access files for screen storage. Requires 8086 or 8088 microcomputer, 64 kbytes RAM, and CP/M-86 operating system.. **$100.00**
With AMD 9511 support routines **$200.00**

MACHINE TEST PROGRAM PACKAGE FOR Z-80 systems. Includes memory, floppy disk, printer, and terminal tests with all source code. Requires CP/M 2.2 .... **$50.00**

All software distributed on eight-inch soft sectored single density diskettes. Prices include shipping by first class or UPS within USA or Canada. COD charges extra. Purchase orders accepted at our discretion. (CP/M and MP/M are registered trademarks of Digital Research, Inc.)

Laboratory Microsystems
4147 Beethoven Street
Los Angeles. CA 90066
(213) 390-9292

This would be done in the form of:

```
if(num==1)
[
        statement_1
]
else
[
        if(num==2)
        [
                statement _2
        [
        else
        [
                if(num==3)
                [
                        statment _3
                ]
        ]
]
```

or if not to complex, as:

```
if(num==1) statement_1
else if(num==2) statements_2
else if(num==3) statements_3
```

Tiny-c has the storage classes of static and extern, but does not support auto, register, or typedef. The compiler supports int, long int, and char data types, but does not have floats, doubles, structs, or unions. Tiny-c has a primitive pointer implementation that will address bytes. Normally, in C, when a pointer is declared an int:

```
int *ptr;
```

it points to an integer boundary and, when incremented, it will pass over that integer and point to the next one (usually two bytes later):

```
++ptr;
```

In tiny-c, the pointer will always increment or decrement one byte, regardless of the data type.

Tiny-c also supports many of the unary and binary operators, with the exception of the unary operators type-name, sizeof, and pointer-to (*ptr), in which the latter is implicit when an array is defined. The binary operators "&&" (connective and), " || " (connective or), "=*", "/=", "%=", "+=", "-=", " << =", " >> =", "&=", " ^ =", and " | =" are also not implemented. The connective and "&&", and or " || " are the most unfortunate losses in that list, not allowing statements such as:

```
if (a==b && c==d) ++c;
```

Tiny-c comes with a wealth of functions in a function library called TCLIB. Most of the relatively common functions listed in standard C are included, plus a set from the tiny-c interpreter known as training functions. These are a set of functions originally set up for beginners, but are a highly convenient set of functions. For instance:

```
pn x            /* Print number x
```

is much more convenient than:

```
fprintf( so, " %d", x)
```

and tend to be very comfortable to use.

Included in the tiny-c TWO package besides the compiler and run-time system is an operating environment called tiny-shell. The tiny-shell is a mised blessing to the compiler. It is loosely based on the UNIX shell developed at Bell Telephone Laboratories. In order to do any work with the tiny-c compiler, you have to invoke the shell from CP/M by typing "sh" to CP/M's Console Command Processor. This puts you into a new command processor that executes the tiny-c run-time package and user-written tiny-c programs. There are some strong disadvantages to the shell as well as very strong advantages. Using the tiny-c compiler at the present time, you cannot execute programs directly from CP/M—you first must enter the shell. This means that you cannot create turnkey programs for resale without an agreement with tiny-c associates, although in speaking with Tom Gibson, I discover that the cost of a resale license is reasonable. In addition to not being able to execute tiny-c programs from CP/M, you also cannot execute CP/M programs from the tiny-shell. If you chose to do some work typing in the programs with a text editor like Word Star, you have to exit the tiny-shell first by typing a Control-C character. This has the habit of displaying a rather unfriendly message to the terminal:

```
ERROR @ 538 ^C
```

Since the tiny-shell is written in tiny-c, it would have been nice to have the input routine check for a Control-C character and gracefully exit the tiny-shell when encountered. In addition, the operator must become familiar with console input control characters that have slightly different actions in the tiny-shell than in CP/M.

Now for the nice features about the tiny-shell. First, for the programmers who like to define the environment they work in, the tiny-shell (as with everything else in the package) is supplied in source code written in tiny-c. This means that you can modify your environment to your heart's content. In addition, the tiny-shell contains some of the features of I/O redirection found in the UNIX shell. For example, you can write a program that will read in a character from the input device, such as the keyboard, and write it to the output device (the screen). When you run it, it will echo everything typed to the screen.

```
%echo
this is a test
^Z
%
```

Simple, right? Now if you were to use redirection of I/O, and typed:

```
%echo <infile.txt >outfile.txt
%
```

you would copy everything in infile.txt to outfile.txt. The indirection characters "< " for input and " >" for output provide this powerful and helpful feature. For instance, you can compile a program and get a listing of lines and errors displayed on the terminal. You may not want to wait until it gets to line 187 to find out what is wrong, and dig through all of that output. The solution is to redirect the output to a disk file. You then run the output file through a text editor or searching program to find what you want, without the hassle of all that printout. An additional tiny-shell feature allows the user to type multiple commands on a line, hit return, and have it execute a series of commands without intervention.

In addition to the functions built into the tiny shell, later versions of tiny-c TWO come with a set of tiny-c programs that can be compiled into UNIX-like shell functions. These include RM which removes (deletes) a file, MV which moves (renames) a file, LS which lists the directory and gives a status display, CAT which concatenates ASCII files, HD which gives a hex dump of all or part of a file, DIFF which is a file comparator, ED which is a UNIX-like line oriented text editor, and APRINT which types a file to the terminal.

The tiny-c TWO package is useful for the person who has learned the tiny-c language with the interpreter and wants to be able to do production work, or for introducing the beginner to structured programming.

The tiny-c TWO package comes with the source to everything including the runtime package in assembler, the compiler, the linker, the function library, the tiny-shell, and shell functions. In addition, it comes with a one inch thick manual bound in a bright white three ring binder. The manual describes the tiny-c language in a way that is clear to beginning users. It then goes into a description of how all of the functions are called, how to interface to the machine language, how to use the compiler and tiny-shell, examples of tiny-c programs, internal details of the compiler, and installation to other operating systems. The sections of the manual describing the tiny-c language are very clear, but the section on bringing up the compiler on other machines appears to be written in a totally different manner that may not be clear to the novice. I found the manual to be too verbose for an experienced programmer, requiring the entire book to be scanned to learn how the package works. It would have been nice to have a chapter summarizing the syntax, function calls, and compiler operation.

The tiny-c TWO package is useful for the person who has learned the tiny-c language with the interpreter and wants to be able to do production work, or for introducing the beginner to structured programming. It is also very good for the experienced programmer who likes to modify his system to suit his needs because of the sources for the tiny-shell and compiler. For the experienced C programmer who doesn't wish to be tied to a non-system environment, or who wants the power of such things as switch/case or a for loop statement, I would recommend one of the compilers designed for standard C.

CP/M C Compilers

	BDS C	Small-C	Tiny-c TWO	Whitesmiths
Version	1.43	1.1(N)	1	2.0
Price	$145	$15	$250	$630
Language	Assembly	C	Tiny-c	C
Minimum System	32K	24K	32K	60K
Format	most	8"SDSS	8"SDSS	8"SDSS

Addresses:

BDS C:

Lifeboat Associates
1651 Third Avenue
New York, NY 10028

Small-C:

The Code Works
Box 550
Goleta, CA 93017

tiny-c TWO:

tiny c associates
P.O. Box 269
Holmdel, NJ 07733

Whitesmiths C:

Whitesmiths, Ltd.
P.O. Box 1132
Ansonia Station
New York, NY 10023

CP/M SOFTWARE SUPERSTARS

MICROPLAN

Throw out your pencils, calculators, worksheets, and perhaps your current computerized worksheet software. MICROPLAN makes them all obsolete. The powerful, flexible, and easy to use features of this decision processing language from Chang Laboratories will please both the dedicated and occassional user. In addition to row and column calculating capabilities, MICROPLAN features four depreciation types, internal rate of return, three loan types, inter-linked multi-staged modeling, programming features, a full featured *report generator,* and *graphics* support. Commands are displayed on the CRT, and processing is determined by menu selection and responses to straight forward questions. Applications include: cashflow planning, budget preparation, pricing analysis, financial report preparation, sales forecasting, . . .

MICROPLAN for CP/M is $495 (order number CL001)

WORDSTAR

Think of a feature that you'd like to see in a word processor and WORDSTAR probably has it. WORDSTAR features full screen editing, automatic margin justification, page formatting, overlapped printing and editing, automatic page numbering, underlining, boldface, subscript, superscript, strikeout, headings, footnotes, variable pitch, proportional printing, online command reference and help facility, and more. With the addition of the MAILMERGE option, WORDSTAR will print personalized form letters, labels, boiler plates, generate legal and other documents, and more. Add the SPELLSTAR option and you'll never make another spelling error. WORDSTAR offers more features than any micro computer word processor we've evaluated. WORDSTAR is a product of MICRO-PRO.

Wordstar $315 (order #MP001) **Spellstar** $160 (order #MP005)
Mailmerge $105 (order #MP002) **Self Instruction Manual** $15 (order #MP004)
Wordstar, Mailmerge, and **Spellstar** $565 (order #MP005)
Wordmaster Program Editor $105 (order #MP006)

TECHNICAL SOFTWARE SYSTEMS

P.O. Box 55, SHREWSBURY, N. J. 07701 (201) 780-4004

N. J. residents add 5% sales tax — VISA and MASTERCARD
Call or write for FREE catalog — Minimum order $100

CP/M is a trademark of Digital Research

Name: Whitesmiths C Compiler
Price: $630
Distributed by:
Whitesmiths, Ltd.
P.O. Box 1132
Anesonia Station
New York, NY 10023

Whitesmiths C Compiler

The Whitesmith C compiler was developed by Whitesmiths, Ltd. of New York City. Their compiler was designed to produce executable programs for the 8080/Z80, LSI-11/PDP-11, VAX-11, M68000. They claim that any program written in C from one machine is portable to any of the other machines, provided that no machine-dependent code is written. This compiler is rather large, and by producing an intermediate code, called A-Natural, can be moved from machine to machine. The A-Natural code (which is assembler-like) can then be compiled/assembled into the executable machine code modules for a given machine.

You could say that the Whitesmiths C compiler for CP/M is the Cadillac of the CP/M C compilers. Like a Cadillac, it is very expensive. At $630, it is the most expensive C compiler available to CP/M. And, as a luxury car eats gasoline, this compiler eats memory. If you don't have 60Kbytes, at minimum, don't even bother to attempt to compile the smallest of C programs. Like the Cadillac, it is cumbersome. A compilation and linkage takes much longer with this compiler than with any others. On the other hand, it tends to generate reasonably fast running code. And also like the Cadillac, it is fancy and has all of the flashy options you could want. The Whitesmiths C compiler complies completely with the "standard," with some extra frills added.

The Whitesmiths package for CP/M comes on two single density floppy disks containing the compiler and machine interface code. In addition to the compilation programs, there is a loader, a library manager, a program to examine relocatable modules, and assembler/compiler to translate the A-Natural code to machine code. There is also a program to translate the A-Natural code into Microsoft Macro Assembler code. In addition to utilities, the Whitesmiths C package comes with a library of 87 high-level functions, seven 8080 C callable subroutines and 55 in-line 8080 subroutines. These final routines are used in the operation of the compiler but can also be used by the systems programmer to do some fancy things like multiplying longs by longs. Finally the package includes twelve functions designed for special calls to the CP/M operating system. At a total of 161 functions available, the programmer has truly large library to chose from.

Also included in the Whitesmiths C package is a set of two rather thick manuals entitled "C Compiler Users Manual" and "C Compiler Systems Interface Manual for 8080 Users." The two manuals are interesting to read. They vary from lucid to something rivaling IBM's most obtuse literature. Only after the third or fourth reading does it become possible to understand some of the things available and some of the things that are not.

The Whitesmiths C compiler complies completely with the standard in Kernighan and Ritchie, including floating point, longs, and storage classes. In addition, it includes some things not yet in the standard. These include types unsigned (char short, long) and character constants with more than one character. It also provides command redirection using the CP/M console command processor, allowing such things as:

```
A>read program.prn >other.prn
```

which would take all output of program.prn and place it in other.prn. This feature adds about 4Kbytes to the generated object code. The following program is 6K and generates code for command redirection:

```
main()
{
}
```

while this next one is 2K and has no redirection code:

```
_main()
{
}
```

Finding out about the above feature was a true exercise in documentation reading. Every programmer should try it once (and only once).

The Whitesmiths C compiler provides most of the standard formatted I/O and file functions, but generally under different names. For instance, the common function "printf" is called "putfmt" in this implementation. It also contains functions that operate on arguments of the command line, many string, numerical and data conversion functions.

Although this C compiler produces reasonably fast code, compilation and linking requires a great deal of patience. It compiles in five steps, generating intermediate files, using the CP/M submit facility. The first step is the preprocessor (pp) followed by the parser (p1) and then the 8080 code generator (p2). Once it has generated the A-Natural source code, it must go through the A-Natural assembler/compiler. Finally it must go through a very long linkage process to bring together all the functions and produce executable code.

This compiler has many tradeoffs, but in certain cases it is very valuable to the system developer. On the negative side are documentation that is not entirely clear, the very long time for edit, compile, test, and edit runs, and its requirement for a huge amount of memory. I had to buy a special chip set for my disk controller just to test this compiler. Also the licensing and ordering processes are very comfortable. After signing a very extensive two page license agreement, I received the compiler. From what I understand from the license agreements, even with the payment of over six hundred dollars, Whitesmiths, Ltd. still owns the thing and can require it to be returned any time in the next fifteen years.

On the positive side, the compiler generates fast code which contains the complete C syntax and is thus portable to other machines. This allows a program, such as a compiler, to be developed on one machine (like the PDP-11) and moved either up to the VAX or down to the 8080/Z80 machines (provided you buy three versions of the compiler, of course).

The Whitesmiths C compiler is definitely not the compiler for a casual user. It is however, quite useful and possibly necessary to the person who needs to compile large, production-type programs using the full C syntax. ■

Plain Talk About Business Computers

Can a Small Computer Really Save You Time?

Time is Money

Theophrastus said time was the most valuable thing a man could spend. Fifteen centuries later Haliburton agreed saying, "we reckon hours and minutes to be dollars and cents." Today, time is more valuable than ever—and more fleeting.

About the only way to gain time is to use it more efficiently and effectively. That's where we come in.

Small Business Computers—by the way, the "small" refers to computers, not to business—will dramatically increase your effectiveness and help save you time and money. How so?

You get flagrantly honest evaluations and reviews of computers and software. We don't just tell you what a program can do; we tell you what it doesn't do, what it does poorly, and what it should do for the price. If advertisers don't like that, we don't want their business, and you're better off without them. Fortunately, most companies appreciate our honesty. In fact, one of our reviewers has gained a reputation because of the many software houses that have incorporated his suggestions into their products. We're proud of that.

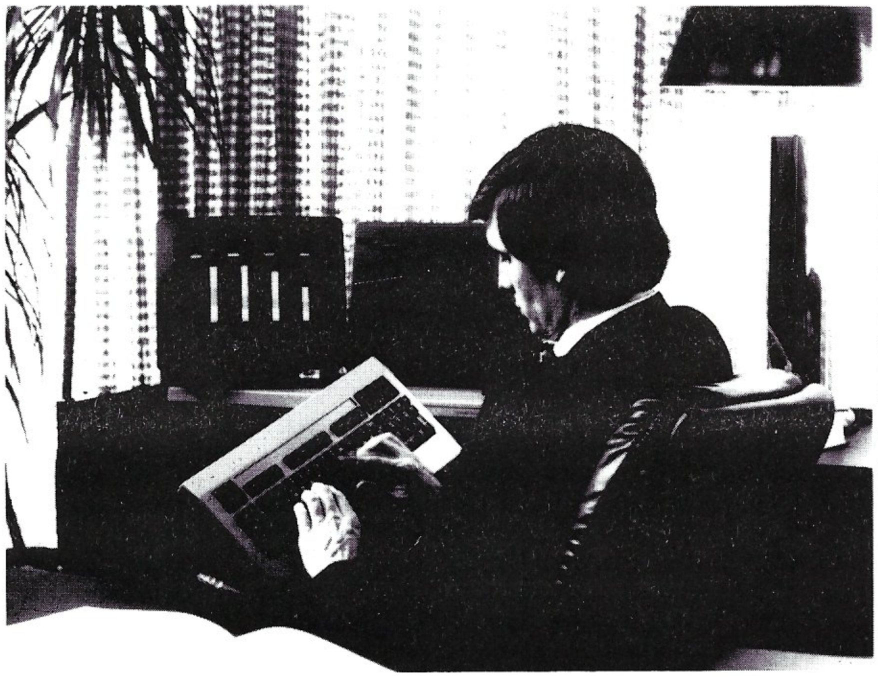


Photo courtesy of Alanthus Data Communications Corp.

Plain Talk

Small Business Computers explains the complexity of today's computerized business world without the technical jargon and doubletalk that may have held you back before. In its easily comprehensible "how-to" style, *Small Business Computers* answers your questions while providing the information you need to make some tough decisions. As you select, purchase, and install your computer system, *Small Business Computers* will guide you through each step calmly and comfortably—helping you to evaluate your computer needs and avoid unnecessary pitfalls. As you use your computer, be it mini or micro, *Small Business Computers* will be there to help you do so efficiently and with confidence while informing you of the latest developments and future possibilities of computers in business.

For Example

You have just purchased a mailing list program. Everything is fine until the file has to be sorted by zip code. If the program has that capability, all is well. If not, you have a big problem. If you had just invested a few hours reading *Small Business Computers*, you would have known what functions to look for before buying the program; you would have known how to plan for future needs. That's just one example. Expand this concept into other areas, other programs and systems, and you can see what you get for your investment.

Added Expertise

As the newest member of the Creative Computing family of fine computer publications, *Small Business Computers* will be expanding to offer subscribers more valuable information than ever before. *Creative Computing* editors and contributors will be unleashing their business expertise in *Small Business Computers* through articles, evaluations and applications of particular interest to the business person. *Creative Computing* has a reputation of editorial excellence and integrity built on unbiased, in-depth product evaluations; articles by top thinkers in the field; and pragmatic, innovative applications.

One management consulting firm, for example, used the Shell-Metzner sort described in *Creative*, and saved $3000 a month, and we still receive letters thanking us for the hardhitting, candid, evaluation of word processing printers we published over a year ago, and which, incidentally, cost us several advertisers.

All this knowledge and experience will now be available to business people in *Small Business Computers*.

So, don't let anyone give you that old story about how complicated and difficult computers are. We don't buy that. Our magazine—our whole philosophy—revolves around the sharing of honest information. If you don't know where to start, we'll put you on the right track. If you're already on the road, we'll show you the best route.

For Any Size Business

Whatever your business—manufacturing or banking, retail or research—*Small Business Computers* will increase your efficiency and help save you time and money.

Subscribe today; *Small Business Computers* is the best consultant your business will ever have.

Order Today

To order your subscription to *Small Business Computers* send $12.00 for 1 year (6 issues). If you prefer, call our toll free number **800-631-8112** (in N.J. 201-540-0445) to put your subscription on your Master Card, Visa, or American Express card. Canadian and other foreign surface subscriptions are $18.00 per year and must be pre-paid. We guarantee that you will be completely satisfied or we will refund the remaining portion of your subscription.

Send orders to:

Small
Business Computers
Magazine

39 E. Hanover Ave.
Morris Plains, NJ 07950
800-631-8112
(In NJ 201-540-0445)

DOS/BIOS Directory And File Conversion In North Star UCSD Pascal (Part II)

by Chris Young

Part II—File Conversion

In this installment we will discuss how to convert the information within the files into standard Pascal data and file types. We will look at some Pascal procedures that facilitate the conversion of North Star data files into similar Pascal data files. This is accomplished by reading the North Star data into standard Pascal types of variables where further processing, including the output of the data to Pascal files, may be done. Three kinds of conversion are discussed: 1) North Star Basic text files into UCSD format text files. 2) North Star Basic string data into Pascal "packed array [..] of char" data. 3) North Star Basic BCD floating point data into Pascal type "real" data.

Text File Conversion

Why should Basic text files be of concern to Pascal users? Prior to the introduction of the new North Word word processing system and Pascal, users did not have a text editing system *specifically* designed for use with North Star systems. True, one can spend over a hundred dollars for CP/M and more for Electric Pencil, Wordstar, or others. However, users of small systems (and users who spent their entire budgets on large systems) may not be able to afford these powerful editors. Even North Word isn't cheap. Many of these users resort to using the Basic program entry editor for their word processing needs. This is accomplished by creating text files full of REM statements. In fact, as long as the user does not try to RUN the files, even the word REM is not necessary. The Basic editor, although limited, can help the user get by on a budget. Most often these files contain documentation of Basic programs, but many other word processing uses can be covered by the Basic editor. A user may have this or many other reasons why he wishes to access Basic text files from Pascal. As his pocketbook recovers, and his editing needs increase, the user may come to the conclusion that for a smaller expenditure for software than is needed for many text editors, he can have the entire UCSD system. This statement is made with the realization that perhaps a significant expenditure for hardware to upgrade to 48K may be necessary. The UCSD system includes a complete Pascal program development system consisting of a compiler, linker, file manager, a Z80 and an 8080 macro assembler, and a powerful screen-oriented text editor. The program described below allows users to access old data files, for whatever reason, for use in this versatile new system.

When a user types a Basic program source text into North Star Basic, the text is not stored exactly as it was entered. Basic recognizes keywords and commands, compacting them into one byte tokens in the range of 128 to 255. The feature not only saves space, but results in faster interpretation of code. When LIST commands are issued, a simple table look-up expands the "crunched" tokens back into their exact original form. However, this space saving is not limited only to Basic statements. Text in string literals and comments (REM statements) are also compacted into one byte tokens wherever possible.

I will now describe the program CVTBAS, found in Listing 4, which performs the function of re-expanding compressed keyword tokens into strings of characters. It creates a UCSD text file or outputs the text to any system device.

First I will examine the procedure GETCHAR which reads the original untyped file, one block at a time, and feeds the characters out one byte at a time. This deblocking routine is the basic tool used in all North Star to UCSD data conversions. GETCHAR comunicates through five variables and an input file which must be declared globally (i.e. at the outermost program level). The file "DOS" is declared as an untyped file (i.e. "var DOS:file;"). LASTCHR is a boolean flag which enables/disables the transmission of characters out of GETCHAR. The flag must be initialized to false at the opening of the input file. After the last byte of the last block has been transmitted, GETCHAR sets LASTCHR to true. Any further calls to GETCHAR returns a null byte. North Star uses a value of one (ASCII character "SOH") as an end of file marker, however,

Chris Young, 3119 Cossell Dr., Indianapolis, IN 46224.

this is context dependent. GETCHAR cannot set LASTCHR true upon reaching a North Star end of file. The program calling GETCHAR can determine if the byte value of one is part of the data or an end of file mark, and it sets LASTCHR is necessary. The character itself is passed to the calling routines in two global variables: CH and CHINT. CH is of type "char" and CHINT is an "integer" whose value is ORD(CH). BUFR is a "packed array [0..511] of char" which is used as the input buffer. BFPTR is an "integer" which points into BUFR. Each time GETCHAR is called, BFPTR is incremented by one. When BFPTR reaches 512, the buffer is empty and a new block is read. To read the first block, the calling program should initialize BFPTR to 512, forcing a block to be read upon the first call to GETCHAR.

The procedure DOLABEL processes Basic labels which are in the form of line numbers. North Star Basic allows line numbers in the range of 1 through 65535 (64K-1). The line numbers are stored as 16-bit <un-signed> integers. Because UCSD Pascal limits integer variables to -32K to +32K, type "real" variables must be used in the calculations which output the line number. Line numbers appear at the beginning of each line and in certain Basic statements. All in-statement label references are preceded by a token code 154. When a code 154 is received at the beginning of each line, the LABL flag is set to true. The causes the invocation of DOLABEL.

DOLABEL uses five variables. LINENUM is of type "real" and is used to store the line number value. P, also type "real," holds a power of ten to be used to strip off the highest order digits one at a time. LEADZERO is a boolean flag, initialized to true upon entry to DOLABEL, which suppresses the output of leading zeros. LEADZERO is reset to false when the first non-zero digit is processed.

DOLABEL, as well as other routines in the system, requires a character to be in CH and CHINT upon entry to the routine. Likewise, all routines also call GETCHAR upon exit, thereby leaving the next character ready for the next section of processing. Upon entry, DOLABEL initializes LEADZERO and puts the low order byte of the line number in LINENUM. Then GETCHAR is called to get the high order byte. This byte is "shifted left" one byte by multiplying it by 256.0 (remember to use "real" arithmetic). Next add in the low order byte and save it in LINENUM. Because we do not want a decimal point output with the value, we do not use the standard WRITE procedure to output the real valued number. Instead, we strip off the high order digits, one at a time, truncate them to one digit integers, and output them to the output device. The loop runs from 4 down to 0 and the variable P is assigned a value of ten to the fourth down to ten to the zero power. This is used to compute what amounts to integer division and modulo calcuations on a real value. As each digit is stripped off, proceeding left to right, the LEADZERO flag is updated and tested. Non-leading zero digits are output to the output device and to the CONSOLE:. The function of the ISFILE flag will be discussed later.

The main program is initialized by two routines: INIT1 and INIT2. Two routines are needed because there is a limit on the size of procedures. The procedures initialize the table as "KEY:packed array [0..255]of string[10]."

C COMPILER
FOR CP/M - 86™

The C86™ Complier implements most features of the C Programming Language. Included is the C language pre-processor, data initialisation, pointers, arrays, structures, longs (64 bits) and floats (64 bits). The complier outputs a relocatable machine language object module.

The package includes a linking loader, a librarian and a large support library. Library functions may be written using either C or ASM-86™. All library source code is included, so that you may extend or modify the library.

$150.00 Disk and Documentation.
$25.00 Documentation only.

Write or call

Computer Innovations, Inc.
75 Pine St., Lincroft, NJ 07738
(201) 530-0995

New Jersey residents include 5% sales tax.
Dealer inquires invited.

CP/M-86 and ASM-86 are trademarks of Digital Research.
C86™ is a trademark of Computer Innovations, Inc.

BASEX
MEANS SPEED!

BASEX is a fast, easy to learn language for 8080, Z80, or 8085 microcomputers. Its commands resemble BASIC, making translation easy. An interactive compiler permits you to enter, list, edit and run programs up to 10x faster than similar BASIC programs and use half the memory (2K plus program).

Powerful features include:

- **Array variables**
- **16 Bit Arithmetic/Logic**
- **Variable name length**
- **Named subroutines with multiple arguments**
- **Text strings**
- **Versatile I/O Functions**
- **Block memory searches/ transfers**
- **Custom commands easily added**

CHOOSE YOUR BASEX . . .

* 97-page BASEX manual, pub. by Byte Books (Includes source listing. Required for use with disks or tapes below.)	$ 8
* Sorcerer/SOL/Poly 88/Meca Alpha/ Heath H8/cassette	$25
* TRS-80® Level II, 16K tape with graphics commands	$25
* North Star Disk, with disk handler commands	$35
* CP/M® Disk, with disk handler commands	$35
* NEW! My Computer Speaks BASEX, pub. by Hayden (a 128-page primer that contains many examples and detailed explanations).	$10

TRS-80 is a trademark of Tandy Corp.; CP/M is a trademark of Digital Research.
Add $ .75 shipping (special 4th class) or $1.50 special handling or UPS.
See BASEX at your local dealer or order direct from

INTERACTIVE MICROWARE, INC.
P.O. Box 771, Dept. State College, PA 16801
Call (814) 238-8294 for SPEEDY ACTION

DEALER INQUIRIES INVITED

The array elements are all first set to a one character string which is the normal ASCII value for that character. Individual elements are then set to strings of characters which will replace the compacted tokens.

After INIT1 and INIT2 are called, the user is prompted to type the name of the Basic text file to be converted. The response is stored in INPNAME and the file is opened by RESET. The user is then prompted for the output file name. If he hits return, the length of OUTNAME is zero, and ISFILE is set false. If a name is entered, the ISFILE flag is set true, and OUTDEV is created as a text file by REWRITE. The text produced by this program is always sent to the CONSOLE: device. If ISFILE is true, the output is also sent to the file OUTDEV.

All lines begin with a length byte which we will skip over. Next, each line has a label. To process the first one, the LABL flag is set to true. A North Star EOF test is made and we enter a "while not LASTCHR do" loop. Another byte is obtained from GETCHAR because all routines require a character to be in CH and CHINT upon entry. IF LABL is true we call DOLABEL and loop back again. Otherwise, a keyword look-up is done by using CHINT as the index into KEY. The string stored in KEY[CHINT] is output. If CHINT is a carriage return (code 13), then we do an extra GETCHAR to ignore the length byte. This is another place where we test for CHINT=1, meaning a North Star EOF was read. Processing continues as above until LASTCHR is true.

Data File Conversion

Users are much more likely to have a need to convert data files from North Star to Pascal formats. These files may be data bases, tables, mailing lists or any one of a number of other types. The overall format of data files is very application dependent. Rather than trying to give samples of specific data conversions, we will present a general conversion program named CVTDATA (See Listing 5). CVTDATA reads a North Star file and outputs its contents to the CONSOLE: device. The program merely demonstrates the techniques needed. The user must adapt the routines and code sections needed to read the North Star data. Once the data is in Pascal variables, further processing may be done on the data, including output to a new file.

Data in North Star disk files is one of four types: an end of file marker one byte long, a floating point number which occupies five bytes in the standard eight digit version, short strings which occupy 1 to 256 bytes, and long strings which occupy more than 256 bytes of space, with a maximum length which is limited by available memory. CVTDATA reads the input file, determines which of these four types it has encountered, and passes the data to the CONSOLE: with a message telling what type of data was read.

String data is processed by the main program and will be covered later. Floating point data is processed in a separate routine and requires some background information before it may be discussed.

North Star floating point data consists of an eight digit binary coded decimal (BCD) normalized mantissa in four bytes, and an exponent in the fifth byte. The high order bit of the exponent byte contains the sign of the mantissa. The remaining seven bits are an excess 64 exponent base 10. A zero exponent means a value of zero for the entire floating point number. UCSD real variables occupy four bytes. There is a 23-bit binary normalized mantissa with an implied leading 1 bit, a mantissa sign bit, and an 8 bit excess 128 exponent base 2. Zero exponents also denote zero value as in North Star.

The user of North Star/ USCD Pascal now has all of the necessary tools to access the contents of DOS/Basic data and text files. With the programs discussed here which allow access to both the directories and the files themselves, the entire package should send the users well on their way to complete conversion to the UCSD system.

What does all of that mean? It means that in North Star, your real variables can have a mantissa plus or minus 9.9999999 and an exponent ten to the plus or minus 64. UCSD has a mantissa plus or minus two to the 24 minus 1, and an exponent two to the plus or minus 128. This works out to an overall range of 9.9999999+64 to -9.9999999E-64 for North Star. UCSD has a range of about 0.1701411247E+37 to -0.1701411247E-37, but only about six to seven digits of the mantissa are significant. Note that the UCSD mantissa can exceed 0.17014—if the exponent is of less magnitude. The magnitude of both the mantissa and the exponent of North Star exceeds that of UCSD. This means that both a loss of significance and a possibility of overflow can occur in North Star to UCSD floating point data conversions.

Exponents on the order of ten to the 32 to ten to the 64 may not concern the user unless his data is of a highly scientific nature. However, the loss of significant digits in the mantissa is of concern in circumstances as every day as a seven digit phone number, or dollar amounts over $10,000.00. The user must deal with these problems on a case-by-case basis.

Real data is converted in CVTDATA by a function called CVTREAL. CVTREAL reads North Star real data through our old friend GETCHAR. It returns a variable real parameter containing the value. A boolean overflow flag is the result of the function. Upon entry to CVTREAL the variable VAL is initialized to zero. Four bytes are read via GETCHAR. Each byte is split into two BCD digits giving eight BCD digits total. VAL is multiplied by ten and a digit is added to it for all eight digits. The last digit or two may not add any significance to the mantissa due to the limitation discussed earlier. Because of the way the mantissa is built up, it will be necessary to divide by 1.0E08 later in the processing. The fifth byte is read and the mantissa sign bit is striped. The excess 64 is subtracted from the exponent, and its sign is determined. Overflow conditions are tested, and if they occur the value of VAL is set to a maximum. The exponent is either multiplied or divided into VAL and VAL is normalized. As usual, GETCHAR is called again to make ready for the next routine.

The main program begins by prompting the user for the input file name. Recall that the program is only a model which outputs to the CONSOLE: device, so no output name is requested. LASTCHR and BFPTR are initialized. GETCHAR obtains the first character.

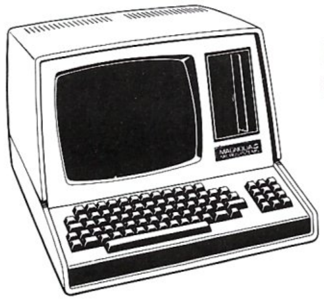
While there are still characters to process, the following actions occur. If CHINT is greater than 15 real data is to be processed. A messages that a number is being processed is printed, and an overflow indication (if necessary) follows that. Finally, the value obtained by CVTREAL is printed. If the original value of CHINT was less than or equal to 15, then a "case" statement is used to process the other options. Option one is an end of file. When North Star EOF is read, a message is printed and LASTCHR is set to true. This causes an exit from the "while" loop and the program terminates. Case two is a long string. Long strings have a two byte, low order byte first, length field. GETCHAR gets the first byte and it is stored in COUNT. GETCHAR gets the second byte, which is multiplied by 256 and added to COUNT. The third case is a short string. The length filed is a one byte field which is read by GETCHAR and saved in COUNT. After leaving the "case" statement, the string is output by a loop whose index runs from 1 to COUNT. The string is output to the CONSOLE: one byte at a time. CHINT values of zero or ten (i.e. nulls and line feeds) are skipped. This is not a necessary feature and can be eliminated if desired. The "one character at a time" output can be replaced by code which puts the characters in a packed array or string for further processing if needed.

The principals demonstrated in CVTDATA can be applied to any North Star data where the precision restrictions are not of concern. Listing 6 is a procedure called WRITREAL which can be incorporated into CVTDATA to replace CVTREAL with slight modification to CVTDATA. WRITREAL reads an eight digit North Star value and outputs it to the CONSOLE: in standard scientific notation. The user may be able to modify this routine to suit the need to extract all of the significance of real data. How he manages to further process this data in Pascal is up to him.

Summary

Some of the many advanced features now available to North Star Pascal users have been demonstrated in the directory conversion programs. The user of North Star/UCSD Pascal now has all of the necessary tools to access the contents of his DOS/Basic data and text files. With the programs discussed above which allow access to both the directories and the files themselves, the entire package should send the users well on their way to complete conversion to the UCSD system. So go to it! ■

Listing starts on page 44.



NEW! for the '89 from

MAGNOLIA MICROSYSTEMS

DOUBLE DENSITY DISK CONTROLLER $595

including CP/M™2.2

This new board adds complete hardware and software support for FOUR 8" Single or Double sided drives and FOUR 5" Single or Double Sided, 48TPI (40 track) or 96TPI (80 track) drives, in addition to the three 5" drives supported by the standard Heath/Zenith controller.

Drive Size	Double Density Capacity
5" Single Sided	162 KBytes
5" Double Sided	343 KBytes
5" 96tpi, Dbl Sided	700 KBytes
8" Single Sided	594 KBytes
8" Double Sided	1210 KBytes

A total **added** capacity of up to 7.6 MegaBytes of on-line storage!

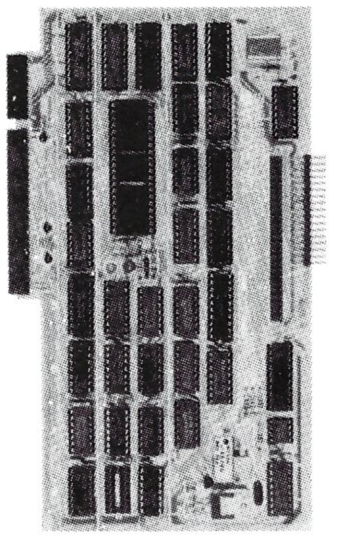
Plus, the obvious advantage of being able to use industry-standard 8" single-density media for program and data interchange.

Full compatibility is retained with existing MMS CP/M support of drives such as the '89's built-in 5" floppy, the Heath/Zenith '47, the REMEX intelligent drive subsystem, the Corvus 10 and 20 MByte Winchester hard disk systems (and Constellation multiplexer), the CAMEO 5+5 Cartridge Disk subsystem with more hard disks soon!

This package includes:
- The Double Density Controller card
- Cables for both 5" and 8" disk drives
- CP/M 2.2 on either 5" or 8" media
- New I/O Decoder and Monitor PROMs

5" and 8" drives are available from us, as well as other suppliers.

If your '89 isn't ORG-0 CP/M compatible yet, our modification is available for $50 additional.



MAGNOLIA MICROSYSTEMS, INC. • 2812 Thorndyke Avenue West
(206) 285-7266 • (800) 426-2841 • Seattle, Washington 98199

CP/M is a trademark of Digital Research.

Listing 4

```
            (* Program to produce a U.C.S.D. text *)
            (*  file from a North Star BASIC text.*)
            (*  Written Aug. '80                  *)
            (*      by Chris Young                *)
            (*         3119 Cossell Drive         *)
            (*         Indianapolis IN 46224      *)
            (*            (317)-291-5376          *)
program CVTBAS;
 var KEY        (* used to map keywords into strings *)
        :packed array[0..255]of string[10];
     CH:char;
     BUFR:packed array[0..511] of char;
     I, J,       (* indicies *)
     CHINT,      (* contains ORD(CH) *)
     BFPTR       (* pointer into input buffer *)
        :integer;
     LASTCHR,    (* is true after last character is processed *)
     LABL,       (* if true, next 2 bytes contain a line number *)
     ISFILE      (* output flag *)
        :boolean;
     OUTNAME,    (* name of output device or file, if null then *)
                 (*  output to console only                     *)
     INPNAME     (* name of BASIC file to be read *)
        :string[20];
     BASIC       (* file of BASIC text to be converted to *)
                 (*   U. C. S. D. text                    *)
        :file;
     OUTDEV      (* file for converted text *)
        :text;
 procedure INIT1;
  begin
   (* initialize array of keyword tokens to default to   *)
   (* their ASCII values                                 *)
   for I:=0 to 255 do
    begin
     KEY[I]:=' ';         (* make strings of length 1 *)
     KEY[I][1]:=CHR(I);  (* put in the ASCII value   *)
    end;
   (* now fill in keywords which are not one-to-one with ASCII *)
   KEY[128]:='LET';      KEY[129]:='FOR';
   KEY[130]:='PRINT';    KEY[131]:='NEXT';
   KEY[132]:='IF';       KEY[133]:='READ';
   KEY[134]:='INPUT';    KEY[135]:='DATA';
   KEY[136]:='GOTO';     KEY[137]:='GOSUB';
   KEY[138]:='RETURN';   KEY[139]:='DIM';
   KEY[140]:='STOP';     KEY[141]:='END';
   KEY[142]:='RESTORE';  KEY[143]:='REM';
   KEY[144]:='FN';       KEY[145]:='DEF';
   KEY[146]:='!';        KEY[147]:='ON';
   KEY[148]:='OUT';      KEY[149]:='FILE';
   KEY[150]:='EXIT';     KEY[151]:='OPEN';
   KEY[152]:='CLOSE';    KEY[153]:='WRITE';
   (* code 154 means a line number label follows *)
   KEY[154]:='';         KEY[155]:='CHAIN';
   KEY[156]:='LINE';     KEY[157]:='DESTROY';
   KEY[158]:='CRFATE';   KEY[159]:='ERRSET';
   KEY[160]:='RUN';
 end; (* INIT2 *)
 (* the INIT procedure is too large so split it in two *)
 procedure INIT2;
  begin
   KEY[161]:='LIST';     KEY[162]:='MEMSET';
   KEY[163]:='SCR';      KEY[164]:='AUTO';
   KEY[165]:='LOAD';     KEY[166]:='CONT';
   KEY[167]:='APPEND';   KEY[168]:='REN';
```

```
   KEY[169]:='NSAVE';    KEY[170]:='SAVE';
   KEY[171]:='BYE';      KEY[172]:='EDIT';
   KEY[173]:='DEL';      KEY[174]:='PSIZE';
   KEY[175]:='CAT';      KEY[176]:='STEP';
   KEY[177]:='TO';       KEY[178]:='THEN';
   KEY[179]:='TAB';      KEY[180]:='ELSE';
   KEY[181]:='CHR$';     KEY[182]:='ASC';
   KEY[183]:='VAL';      KEY[184]:='STR$';
   KEY[185]:='NOFNDMARK';       KEY[186]:='INCHAR$';
   KEY[187]:='FILE';     KEY[224]:='(';
   KEY[255]:=';        KEY[226]:='*';
   KEY[227]:='+';        KEY[228]:='[';
   KEY[229]:='-';        KEY[231]:='/';
   KEY[236]:='AND';      KEY[237]:='OR';
   KEY[239]:='>=';       KEY[240]:='<=';
   KEY[241]:='<>';       KEY[244]:='<';
   KEY[245]:='=';        KEY[246]:='>';
   KEY[247]:='NOT';      KEY[198]:='INT';
   KEY[204]:='LEN';      KEY[205]:='CALL';
   KEY[206]:='RND';      KEY[202]:='SGN';
   KEY[203]:='SIN';      KEY[210]:='ATN';
   KEY[216]:='FREE';     KEY[217]:='INP';
   KEY[218]:='EXAM';     KEY[219]:='ABS';
   KEY[220]:='COS';      KEY[221]:='LOG';
   KEY[222]:='EXP';      KEY[223]:='TYP';
 end; (* INIT2 *)
 procedure GETCHAR;
  (* This procedure gets a character from the file BASIC and    *)
  (* returns the character in the global variables CH and CHINT.*)
  var N:integer;
  begin(*GETCHAR*)
   if not LASTCHR        (* if last character has *)
                         (* been read, don't try again *)
    then begin
          if BFPTR=512  (* buffer is empty, get new buffer *)
           then if EOF(BASIC)
                 then begin     (* no more buffers in file *)
                       BFPTR:=0;
                       BUFR[0]:=CHR(0);
                       LASTCHR:=true; (* this shuts off GETCHAR *)
                                      (* even if N* EOF has not *)
                                      (* been read              *)
                      end (* if EOF(BASIC) *)
                else begin
                      N:=BLOCKREAD(BASIC,BUFR,1);
                      BFPTR:=0;
                     end; (* if not EOF(BASIC) *)
          CH:=BUFR[BFPTR]; CHINT:=ORD(CH);
          BFPTR:=BFPTR+1;
         end; (* if not LASTCHR *)
  end; (* GETCHAR *)
 procedure DOLABEL;
  (* This procedure is called after every occurence of the       *)
  (*  label flag which is code 154, and at the begining of every *)
  (*  line. It reads 2 bytes and outputs the integer value as a  *)
  (*  string of ASCII characters.  The value is interpreted as a *)
  (*  16 bit unsigned integer from 0 to 65535.  Note this is     *)
  (*  outside the range of Pascal integer variables, so real     *)
  (*  variables are used.                                        *)
  var LINENUM,   (* the value of the label as a real number      *)
      P          (* real power of ten  *)
          :REAL;
      LEADZERO  (* supresses output of leading zeroes    *)
          :boolean;
  begin
   LEADZERO:=true;
```

ZAP! typos and spelling errors with

The Disc-tionary

Documents with spelling errors and typos can destroy your company's reputation. The Disc-tionary allows you to find and correct errors quickly and easily.

Recognizes more than 40,000 words!

No other spelling correction system has this unique combination of ease of use, versatility, and value.

Ease of use

The Disc-tionary is not a collection of utility programs. The text is scanned at amazing speed, checked against the dictionary, and optionally marked by a single program.

When an unrecognized word is found, it may be added, rejected, ignored, or changed. Also, similar words may be displayed.

All commands are selected from menus with the touch of a single key.

Dictionary editing, word counting, and other functions are included in the main program.

Versatility

Optional modes are provided for automatic addition or rejection of all unrecognized words.

Hyphens, and apostrophes are correctly handled.

Optionally removes dozens of suffixes automatically (with or without user verification) or manually. Options allow total control of suffix removal.

Large expansibility.

Value

$79.00 complete

At **$79.00**, The Disc-tionary is priced far below similar programs and only a few dollars more than far less sophisticated ones.

If you're not convinced, the manual is available separately at **$15.00**. (deductable from software price.) Software price includes 2 free "bug fix" updates - if you ever need them.

No shipping charge on prepaid orders. Mich. residents add 4%.

Requires CP/M*, 8 inch drive, Z80* CPU and 32K RAM. Works with ED, WordStar* and most other editors and word processors. (Call or write for compatibility details.)

*Trademarks: WordStar - MicroPro, CP/M - Digital Research, Z80 - Zilog.

STELLARSOFT

841 Blanchette Dr., East Lansing, MI. 48823. Phone (517) 332-2459

North Star
BASIC UTILITY SET

- EDITOR — Create & edit a Basic program using 26 commands, including GLOBAL locate & change.
- BPRT — Print & cross reference a Basic program.
- BPAK — Pack a Basic program.
- RE — Rename a disk file.

$69 plus $1.50 shipping. Calif. Res. add 6%. Check, VISA, M.C

SZ Software Systems

1269 Rubio Vista Road, Altadena, Calif. 91001
(213) 791-3202

CP/M® ↔ IBM
CP/M ↔ DEC
Compatibility with
REFORMATTER™

Exchange data files with most IBM and DEC equipment through ***REFORMATTER*** disk utilities. With ***REFORMATTER***, you can read and write IBM 3740 and DEC RT-11 formatted diskettes on your CP/M system. Programs feature bi-directional data transfer and full directory manipulation. ASCII/EBCDIC conversion provided with CP/M↔IBM.

Each program $195.00 from stock. Specify CP/M↔IBM or CP/M↔DEC when ordering.

Program Data Sheets and Application Guide available from MicroTech Exports, Inc., 467 Hamilton Ave., Suite 2, Palo Alto, CA 94301 □ Tel: 415/324-9114 □ TWX: 910-370-7457 MUH-ALTOS □ Dealer & OEM discounts available.

CP/M® is a registered trademark of Digital Research.

INFOSOFT Has a Better Way

MultI/OS™

THE CP/M™ COMPATIBLE WORLD ENTERS A NEW ERA!!

Call Us Today To Improve Your System Performance

InfoSoft
SYSTEMS INCORPORATED
25 SYLVAN ROAD SOUTH
WESTPORT, CONN. 06880
(203) 226-8937

CP/M is a registered trademark of Digital Research CA

```
   LINENUM:=CHINT;       (* low order byte  *)
   GETCHAR;              (* high order byte *)
   LINENUM:=LINENUM + 256.0 * CHINT;
   LABL:=false;          (* am no longer reading label *)
   for I:=4 downto 0 do
    begin
     P:=PWROFTEN(I);
     J:=TRUNC(LINENUM/P);        (* divide out high order digit *)
     LINENUM:=LINENUM - J * P;
     LEADZERO:=LEADZERO and (J=0);    (* LEADZREO true until J<>0 *)
     if not LEADZERO then
      begin
       if ISFILE then WRITE(OUTDEV,J);
       WRITE(J);
      end; (* if not LEADZERO *)
    end; (* for I *)
  end; (* DOLABEL *)
 begin (* MAIN *)
  INIT1;  INIT2;
  WRITE('Type BASIC File-name:'); READLN(INPNAME);
  RESET(BASIC,INPNAME);
  WRITE('Type new file name:');   READLN(OUTNAME);
 if LENGTH(OUTNAME)>0 then begin
                              REWRITE(OUTDEV,OUTNAME);
                              ISFILE:=true
                             end
                        else ISFILE:=false;
  LABL:=true;    (* file starts with a label *)
  LASTCHR:=false;
  BFPTR:=512;    (* this means buffer is empty *)
  GETCHAR;       (* skip length byte *)
  LASTCHR:=LASTCHR or (CHINT=1);      (* test for N* EOF *)
  while not LASTCHR do
   begin
    GETCHAR;
    if LABL
     then DOLABEL
     else begin
           LABL:=(CHINT=154) or (CHINT=13);
           if ISFILE then WRITE(OUTDEV,KEY[CHINT]);
           WRITE(KEY[CHINT]);
           if CHINT=13 then
            begin
             GETCHAR;        (* skip length byte *)
             LASTCHR:=LASTCHR or (CHINT=1);   (* test for N* EOF *)
            end;(*if CHINT=13*)
          end;(*if LABL*)
   end;(*while*)
  CLOSE(OUTDEV,LOCK);
 end.
```

Listing 5

```
                (* Program to demonstrate North Star  *)
                (*  to U.C.S.D. Pascal data conv.     *)
                (*  Written Aug. '80                  *)
                (*      by Chris Young                *)
                (*          3119 Cossell Drive        *)
                (*          Indianapolis IN 46224     *)
                (*             (317)-291-5376         *)
program CVTDATA;
 var CH         (* character returned by GETCHAR *)
         :char;
     BUFR       (* input buffer used by GETCHAR   *)
         :packed array[0..511] of char;
     I, J,      (* indicies *)
     CHINT,     (* contains ORD(CH) returned by GETCHAR *)
     COUNT,     (* length of string *)
     BFPTR      (* pointer into input buffer *)
         :integer;
     LASTCHR    (* is true after last character is processed *)
         :boolean;
     VALU       (* temp real *)
         :real;
     DOS        (* N* DOS format data file for input *)
         :file;
     INPNAME    (* name of DOS file to be read *)
         :string[20];
procedure GETCHAR;
 (* This procedure gets a character from the file DOS and *)
 (*   returns the character in the global variables *)
 (*   CH and CHINT.           *)
 var N:integer;
 begin(* GETCHAR *)
  if not LASTCHR         (* if last character has *)
                         (* been read, don't try again *)
   then begin
         if BFPTR=512  (* buffer is empty, get new buffer *)
          then if EOF(DOS)
                then begin     (* no more buffers in file *)
                      BFPTR:=0;
                      BUFR[0]:=CHR(0);
                      LASTCHR:=true; (* this shuts off GETCHAR *)
                                     (* even if N* EOF has not *)
                                     (* been read              *)
                     end (* if EOF(DOS) *)
                else begin
                      N:=BLOCKREAD(DOS,BUFR,1);
                      BFPTR:=0;
                     end; (* if not EOF(DOS) *)
         CH:=BUFR[BFPTR]; CHINT:=ORD(CH);
         BFPTR:=BFPTR+1;
        end; (* if not LASTCHR *)
 end; (* GETCHAR *)
function CVTREAL(var VAL:real): boolean;
  var EXPSGN,  (* sign of exponent, +1 or -1    *)
      HI,      (* high order 4 bit BCD value of a byte *)
      LO,      (*  low order 4 bit BCD value of a byte *)
      I,       (* index *)
      EXPON    (* exponent as integer *)
          :integer;
      A        (* real value of ten to the EXPON *)
          :real;
    OVFL:boolean;
begin
 VAL:=0;
 for I:=0 to 3 do    (* read 4 bytes, 8 BCD digits *)
  begin
   HI:=CHINT div 16; (* high order 4 bits *)
   LO:=CHINT mod 16; (*  low order 4 bits *)
   VAL:=VAL * 100 + HI * 10 + LO;    (* build up value *)
   GETCHAR;
  end;
 if CHINT>127 then   (* high order bit of 5th byte *)
                     (*  is sign of mantissa       *)
    begin
     CHINT:=CHINT-128;
     VAL:=-VAL;
    end;
   EXPON:=CHINT;
   if EXPON<>0          (* zero exponent means zero value *)
    then
     begin
      EXPON:=EXPON-64; (* exponent is excess 64 *)
      if EXPON>0 then EXPSGN:=1 else EXPSGN:=-1;
      EXPON:=ABS(EXPON);
      (* Note:  0.1701411247E+-37 is the limit before overflow *)
      OVFL:=(EXPON>37)or((EXPON=37)and(VAL>17014112.0));
      if OVFL then begin EXPON:=37; VAL:=17014112.0 end;
      A:=PWROFTEN(EXPON);
      if EXPSGN<0 then A:=1.0/A;
      (* VAL is on the order of 1E8 so normalize it and *)
      (*  multiply in exponent.                         *)
      VAL:=VAL/1.0E8 * A;
     end (* if EXPON<>0 *)
    else
     begin
      VAL:=0; OVFL:=false;
     end;(*if EXPON=0*)
   GETCHAR;     (* always leave a character in CH and CHINT *)
   CVTREAL:=OVFL;
  end;(*CVTREAL*)
  begin (* CVTDATA *)
   WRITE('Type DOS data file name:'); READLN(INPNAME);
   RESET(DOS,INPNAME);
   LASTCHR:=false;
   BFPTR:=512;   (* this means buffer is empty *)
   GETCHAR;      (* skip length byte *)
   LASTCHR:=LASTCHR or (CHINT=1);        (* test for N* EOF *)
   while not LASTCHR do
    begin
     if CHINT>15 (* this means it is a real value *)
      then begin
            WRITE('Number =');
            if CVTREAL(VALU) then WRITE('**** OVERFLOW **** ');
            WRITELN(VALU);
           end (* if CHINT>15 *)
    else
       begin
        case CHINT of
         1: begin
             WRITELN('EOF');
             LASTCHR:=true;
            end;
         2: begin
             WRITE('Long string ');
             GETCHAR;
             COUNT:=CHINT;
             GETCHAR;
             COUNT:= 256 * COUNT + CHINT;
             GETCHAR;     (* always leave a character *)
                          (*   in CH and CHINT         *)
            end;
         3:begin
            WRITE('Short string ');
            GETCHAR;
            COUNT:=CHINT;
            GETCHAR;      (* always leave a character *)
                          (*   in CH and CHINT         *)
           end;
        end; (* case *)
       if not LASTCHR then
        begin
         WRITE('"');
         for I:=1 to COUNT do
          begin
           if (CHINT<>0) and (CHINT<>10) then WRITE(CH);
           GETCHAR;
          end; (*for I:=1 to CHINT *)
         WRITELN('"');
        end; (* if not LASTCHR *)
      end; (* if CHINT>15 then ... else *)
   end; (* while not LASTCHR *)
 end. (* CVTDATA *)
```

Listing 6

```
                (* Program to output North Star 8 dig.*)
                (*  real to CONSOLE:                  *)
                (*  Written Aug. '80                  *)
                (*      by Chris Young                *)
                (*         3119 Cossell Drive         *)
                (*         Indianapolis IN 46224      *)
                (*            (317)-291-5376          *)
procedure WRITREAL;
(* Reads North Star 8 digit real and outputs it to CONSOLE:   *)
 var I,        (* index *)
     EXPON     (* exponent as integer *)
         :integer;
     MANTSGN   (* if true, mantissa is negative *)
         :boolean;
     DIGITS    (* array of 8 digits *)
         :packed array[0..7] of char;
 begin
  for I:=0 to 3 do    (* read 4 bytes, 8 BCD digits *)
   begin
    DIGITS[I*2]:=CHR((CHINT div 16)+48);    (* high order 4 bits *)
    DIGITS[I*2+1]:=CHR((CHINT mod 16)+48); (*  low order 4 bits *)
    GETCHAR;
   end;
  if CHINT>127  (* high order bit of 5th byte is sign of mantissa *)
   then
    begin
     CHINT:=CHINT-128;
     MANTSGN:=true;
    end
   else MANTSGN:=false;
  EXPON:=CHINT;
  if EXPON<>0         (* zero exponent means zero value *)
   then
    begin
     EXPON:=EXPON-64; (* exponent is excess 64 *)
     if MANTSGN then WRITE('-') else WRITE('+');
     WRITE('0.');        (* output first digit and decimal pt. *)
     for I:=0 to 7 do WRITE(DIGITS[I]);  (* output digits *)
     WRITE('E',EXPON);
    end (* if EXPON<>0 *)
   else
    WRITE('0.000000E+00');
  GETCHAR;     (* always leave a character in CH and CHINT *)
 end;(*WRITREAL*)
```

Virtual Segment Procedures under UCSD Pascal

by Jon Bondy

One of the nice features of UCSD Pascal is its support of segment procedures. A segment procedure is like any other Pascal procedure except that whenever it is called (except for recursive calls), it is loaded from disk memory prior to being executed; and after it exits, the memory is reclaimed. In fact, the segment procedure is loaded onto the stack since the pattern of memory use is nested in a very stack-like manner.

A segment procedure is like any other Pascal procedure except that whenever it is called (except for recursive calls), it is loaded from disk into memory prior to being executed; and after it exits, the memory is reclaimed.

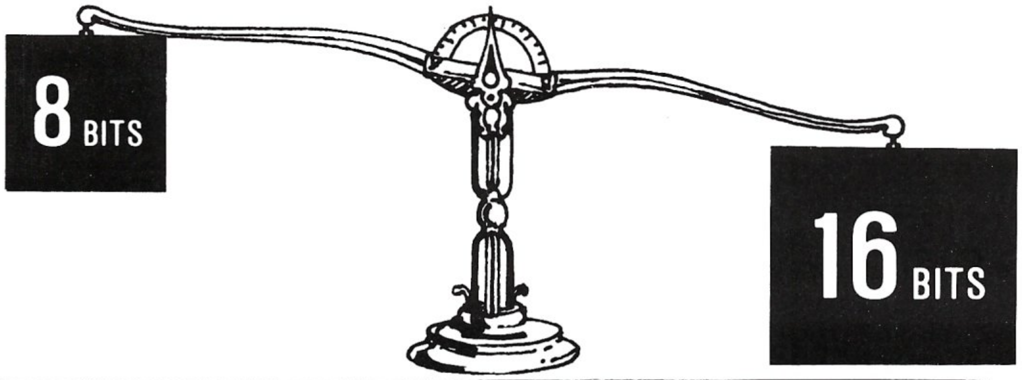
Segment procedures allow the programmer to manage memory resources explicitly and conveniently, and really are a form of overlay. In large programs, it is not uncommon to dedicate a segment procedure to initialization, since that code need not reside in memory after the program starts. Segment procedures may have internal procedures and functions, all of which are loaded with the segment procedure code, allowing functional groups of routines to be brought into memory in a single operation, and executed.

Jon Bondy, Box 148, Ardmore, PA 19003.

Segment procedures allow the programmer to manage memory resources explicitly and conveniently, and really are a form of overlay.

Unfortunately, UCSD Pascal allows the programmer access to only six segment procedures under normal circumstances. This is fine for small programs (under 3000 lines), but when one starts to get serious about an application, more segment procedures are necessary. One reason for this is that the UCSD Pascal separate compilation construct (UNITs) uses one of these six segment procedure slots even if it is not loaded into memory dynamically, wasting this scarce resource. This situation should improve very soon (perhaps by the time this article is printed) because Softech is planning to announce features in UCSD Pascal Version IV.O to solve some of these problems.

I work for a company (Energy Data Systems) which has been trying to do some fairly complex applications in UCSD Pascal, and we ran into the "segment barrier" in the spring of 1980. We considered modifying the UCSD operating system since we had source for it, but decided to try to stick to solutions which required as few



MICROCOMPUTERS

THE CHOICE JUST GOT EASIER!

PRICE $4390 Gets you a Tec-86 Complete 16 Bit 8086 System, including 64K Bytes of Memory, Dual 8 Inch Double Density Floppy Disks, and more. . . (See complete Tec-86 specs below.)

PERFORMANCE Faster Processing, Increased Address Space, up to One Megabyte of available memory, and the Increased throughput of the the TEC-86 16 Bit System means that you can now do work on a microcomputer never before practical with the slower 8 Bit systems.

SOFTWARE CP/M-86™ Brings the World of CP/M software up to the 16 bit systems. BASIC-86™, MS-Fortran™, MS-PASCAL™ are just a few of the Languages currently available.

EXPANSION Memory Expansion up to ONE MEGABYTE. Disk Storage can be increased to over 30 Megabytes with our Winchester Disk Subsystem. Multi User Systems available soon!

QUALITY Full ONE YEAR WARRANTY is standard on TECMAR components. This Warranty is possible due to our High Engineering Standards, and Thorough Product Testing.

DELIVERY We have been delivering 16 Bit Boards and Systems for nearly two years. Most orders are shipped from stock or within 30 days.

S100 A/D $765
Analog to Digital Converter with Timer/Counter

12 Bit Accuracy, 16 Channels, 30 KHz Conversion, Five Programmable Timer/Counters for:
- Time of Day
- Triggering Conversions
- Counting Conversions
- Counting External Signals
- Interrupting the CPU

Options: Programmable Gain; 14, 16 Bit Accuracy; 40, 100, 125 KHz Conversion; Expansion to 256 Channels

$4390 TEC-86 $4390
16 BIT 8086 MICROCOMPUTER SYSTEM

FEATURES: 5 MHz 8086 CPU * 64K Bytes RAM * S100 IEEE-696 * Dual 8 Inch Double Density Floppy Disk Drives * Two RS 232 Serial Ports, 50 to 19200 Baud * Three Parallel Ports * ROM Boot for CP/M-86 * Heavy Duty Power Supply * Attractive Industrial Quality Desk Top Cabinet * Completely Assembled and Tested * **OPTIONS:** Up to One Megabyte of Memory, Double Sided Double Density Floppy Disks * Up to 31 Megabytes of Winchester Fixed Disk Storage. * CP/M-86 * BASIC-86 * MS-FORTRAN * MS-PASCAL * CBASIC/86™ * PASCAL/M86™

$10990 TEC-86W $10990
Same Great Features as our TEC-86 except that the TEC-86W has 256K of RAM, and our 31 Megabyte Winchester Disk Subsystem.

NEW — 64K/256K S100 MEMORY $595/$1695

8 and 16 Bit Transfer for 8 and 16 Bit Systems
MP/M and MP/M-86 Compatible, Up to 7 users on one 256K board.
Parity Detection with Interrupt and LED Indication
Hardware Write Protect
Full DMA Capability
24/20/16 bit Address Decoding
Enable/Disable 8K/32K Segments
Bank Select in 8K/32K Segments
Meets IEEE-696 Specifications

See our catalog for many other fine S-100 and Apple Boards including: — Analog to Digital Converters *(16-256 channels; 12, 14, or 16 bit accuracy; 30, 40, 100, or 125 KHz; programmable gain; timer/counters)* **— Digital to Analog Converters** *(12 bit accuracy, 3 microsecond conversion)* **— 8086 CPU Board — I/O Boards — 16K Static RAM — Real-time Video Digitizer and Display.**

CP/M MP/M CP/M-86 and MP/M-86 are trademarks of Digital Research Inc. BASIC-86, MS FORTRAN and MS-PASCAL are trademarks of MicroSoft Inc. PASCAL/M86 is a trademark of Sorcim CBASIC/86 is a trademark of Compiler Systems Inc.

TECMAR INC.

DEALER INQUIRIES INVITED

23600 Mercantile Rd. • Cleveland, OH 44122

TECMAR, INC.
(216) 464-7410

modifications to operating system code as possible. After some thought, I came up with an interim means of ameliorating the problem, which I will describe in this article.

When a segment procedure is called by the UCSD Pascal p-machine, a special op-code is used. This op-code first looks in an operating system segment table to see if the indicated procedure has been called more than zero times (reference count, in case it is a recursive call and the code need not be reloaded). If it is already in memory (reference count > 0), it is executed like any other procedure; if not, the op-code looks in the segment table for the block number on the disk where it can find the code for the segment procedure, and the number of bytes in the code. It then loads the code onto the stack and calls the procedure, incrementing the reference count to one. Upon exit, the segment procedure return op-code decrements the reference count by one, and clears the stack back up to where it was prior to the call if the count has become zero.

My idea was to somehow write data into the segment tables prior to each segment procedure call in such a way that when the above op-code was invoked, it would find data describing different segment procedures each time; it would in fact be faked into loading different code segments into memory for each call, even though the same segment procedure was being called each time.

In order to do this, I first had to find where the segment tables were located in memory. The first global variable declared in the UCSD Pascal operating system code is a pointer to a special record, called the system communication record, or SYSCOMREC. The segment tables are a part of this record. There is a special switch in the UCSD Pascal compiler (the 'U' switch) which, when its value is '-', causes programs to function very differently than usual. The main program does not execute at all, but rather the first segment procedure declared in the program executes instead. Also, the variable definitions are "aliased" on top of the definitions for the operating system (like an EQUIVALENCE in Fortran), allowing the program to read and write those variables. Usually, one uses exactly the same variable definitions as are used when the operating system is compiled, in order that the program agree with operating system definitions. In this case, however, I simply made my own definitions, since all I wanted was to determine the value of the pointer to SYSCOMREC. Since 1.5 stores pointers as actual memory addresses, by printing the value of the pointer I could determine where in memory SYSCOMREC was stored.

I wrote the following program and was able to locate SYSCOMREC. It was part trial and error, since after I

```
(*$U-*)
program find;
  var
    i : integer; (* aliased to ↑syscomrec *)

  segment procedure findsyscom;
    begin
    writeln('Syscomrec is located at address ',i,' decimal.');
    end;

begin
end.
```

thought I had found SYSCOMREC the first time, I was forced to look through dumps to figure out why I had been wrong. Anyway, the program below will tell you where in memory any SYSCOMREC is under Version I.5 (it is at location 718 [decimal] for my Z-80 version of I.5), and probably under Version II.0. I have not tried it with III.0.

I then needed to figure out how far from the start of SYSCOMREC the segment tables started. Fortunately, the UCSD Pascal operating system variable definitions, found in a file called GLOBALS.TEXT, were distributed with UCSD Pascal Versions 1.4 and 1.5, so I had them at my disposal. [The GLOBALS.TEXT file, along with all other UCSD Pascal source code, is a copyright of the University of California at San Diego; some of the GLOBALS.TEXT file is presented in this article (the portions of Pascal source in capital letters) with the permission of Softech Microsystems Inc., their licensee.] I am informed that the SYSCOMREC data layout has not changed with the various UCSD Pascal Versions, so you should be able to locate the segment tables at 96 bytes past the start of your SYSCOMREC. This means that in writing to location 814 in my memory, one would be writing on the first byte of the segment tables.

The segment tables are defined as follows:

```
type

  segdesc = record
    diskaddr : integer; (* absolute block number on disk *)
    codeleng : integer; (* in bytes *)
    end;

  segtab = array [segrange] of record
    unit : unitnum; (* disk unit number (an integer) *)
    codedesc : segdesc; (* as above *)
    end;
```

where "segrange" is the number of segment procedures defined for the UCSD system being used ("0..15" in the case of Version 1.5). If one declared a pointer "segptr" which pointed to a record of type "segtab", one could refer to that record as "segptr ↑", to the unit (disk drive) on which the code for the third segment procedure was located as "segptr↑ [3].unit", and to the number of bytes in the code for that procedure as "segptr↑ [3].codedesc. codeleng". One could establish the correct value in that pointer with the following record definition:

```
var

  alias : record case boolean of
    true : (i : integer);
    false: (p : ↑segtab);
    end;
```

This record definition states that one will either use the storage for the record alias as an integer (denoted "alias.i") or as a pointer to a variable of type "segtab" (denoted "alias.p"). Since the same storage is used for both values, if one were to write into the integer part, one could then use that value as a pointer; one could trick Pascal into thinking that a variable of type "segtype" was being pointed to.

By stating "alias.i := 718 + 96", I could access the actual operating system segment table as "alias.p ↑". I could then write into the segment table entries of any segment which I chose, forcing the system to load the code I wanted to when I called the appropriate procedure.

Suppose, for instance, that I knew that I wanted to execute each of a series of ten segment procedures which were located on "unit[i]", had length "length[i]" bytes, and started at disk block number "block[i]". I could write a procedure to perform a call to the "i-th" such segment procedure as follows:

```
program test;

   type
     segdesc = record
       diskaddr : integer; ( absolute block number on disk )
       codeleng : integer; ( in bytes )
       end;
     segtab = array [segrange] of record
       unit : unitnum; ( disk unit number )
       codedesc : segdesc; ( as above )
       end;

   var
     alias : record case boolean of
       true : (i : integer);
       false: (p : ↑segtab);
       end;
     unit : array[1..10] of integer;
     length : array[1..10] of integer;
     block : array[1..10] of integer;
     i : integer;

   segment procedure virtual;
     begin ( need not have any code, since it will never execute --
             the other ten segment procedures will execute instead )
     end; ( virtual )

   procedure dovirtual(i : integer);
     begin
     ( set up to call virtual segment )
     alias.p↑[10].codeunit := unit[i];
     alias.p↑[10].codedesc.codeleng := length[i];
     alias.p↑[10].codedesc.diskaddr := block[i];
     ( call virtual segment loaded above )
     virtual;
     end; ( dovirtual )

   begin
   ( set up pointer to real segment table )
   alias.i := 718 + 96;
   ( call the ten virtual procedures )
   for i := 1 to 10 do dovirtual(i);
   end. ( test )
```

The above program will work, but has a few problems which make it a bit awkward to use. First, procedure calls which previously looked like a nice name now are reduced to a cryptic statement like "dovirtual(3)". This problem can be corrected by creating constants at the start of the program with values from 1 to 10, and calling "dovirtual" with those constant values; a procedure to clear the screen might then be called as "dovirtual (clrscreen)", a significant improvement.

The other problem is that it is *not easy* to find out some of the information which I so casually stated would be found in the length and block arrays. To do so involves reading the segment table of the code file in which one of the virtual segments exists. Unfortunately, the disk address information stored in the segment tables of a code file is slightly different than that stored in the operating system's segment tables. The normal code file disk addresses are relative to the start of the entire code file; the system addresses are the absolute disk block number. This means that in order to convert the data in the code file's segment table into useful information, we must also know the absolute address of the start of the code file. And in order to determine this, we must read (and understand) the directory of the disk. Whew!...

Taking it a step at a time, the format of a UCSD Pascal disk directory is given below. It is a portion of the UCSD GLOBALS.TEXT file mentioned earlier (and is copyrighted by UCSD).

```
CONST
  MAXUNIT = 12;          (*MAXIMUM PHYSICAL UNIT # FOR UREAD*)
  MAXDIR = 77;           (*MAX NUMBER OF ENTRIES IN A DIRECTORY*)
  VIDLENG = 7;           (*NUMBER OF CHARS IN A VOLUME ID*)
  TIDLENG = 15;          (*NUMBER OF CHARS IN TITLE ID*)
  FBLKSIZE = 512;        (*STANDARD DISK BLOCK LENGTH*)
  DIRBLK = 2;            (*DISK ADDR OF DIRECTORY*)
  MAXSEG = 15;           (*MAX CODE SEGMENT NUMBER*)

TYPE

  DATEREC = PACKED RECORD
    MONTH: 0..12;              (*0 IMPLIES DATE NOT MEANINGFUL*)
    DAY: 0..31;                (*DAY OF MONTH*)
    YEAR: 0..100               (*100 IS TEMP DISK FLAG*)
    END (*DATEREC*) ;

  UNITNUM = 0..MAXUNIT;
  VID = STRING[VIDLENG];

  DIRRANGE = 0..MAXDIR;
  TID = STRING[TIDLENG];

  FILEKIND = (UNTYPEDFILE,XDSKFILE,CODEFILE,TEXTFILE,
    INFOFILE,DATAFILE,GRAFFILE,FOTOFILE,SECUREDIR);

  DIRENTRY = RECORD
    DFIRSTBLK: INTEGER;                    (*FIRST PHYSICAL DISK ADDR*)
    DLASTBLK: INTEGER;                     (*POINTS AT BLOCK FOLLOWING*)
    CASE DFKIND: FILEKIND OF
      SECUREDIR,
      UNTYPEDFILE:                         (*ONLY IN DIR[0]...VOLUME INFO*)
        (DVID: VID;                        (*NAME OF DISK VOLUME*)
         DEOVBLK: INTEGER;                 (*LASTBLK OF VOLUME*)
         DNUMFILES: DIRRANGE;              (*NUM FILES IN DIR*)
         DLOADTIME: INTEGER;               (*TIME OF LAST ACCESS*)
         DLASTBOOT: DATEREC);              (*MOST RECENT DATE SETTING*)
     XDSKFILE,CODEFILE,TEXTFILE,INFOFILE,
     DATAFILE,GRAFFILE,FOTOFILE:
        (DTID: TID;                        (*TITLE OF FILE*)
         DLASTBYTE: 1..FBLKSIZE;           (*NUM BYTES IN LAST BLOCK*)
         DACCESS: DATEREC)                 (*LAST MODIFICATION DATE*)
    END (*DIRENTRY*) ;

var
  directory : array[dirrange] of direntry;
```

Given the above definitions, after a bit of studying we can see that each directory entry contains the disk address (in blocks) of the first and last blocks of the file for which it is an entry. If the entry is a volume ID entry, it also contains the volume (diskette) name, the number of blocks on the volume (deovblk), the number of files on the volume (dnumfiles) and the time (date) when it was last accessed. Normally one finds this entry as the first entry in the directory (or "directory[0]"). If the entry is a normal file entry, it contains the file ID (name), the number of bytes in the last block which really contain data, and the date of the last modification to the file. The above definitions also tell us where to find the directory, namely at "dirblk", or block two.

We can read the directory into memory with the statement

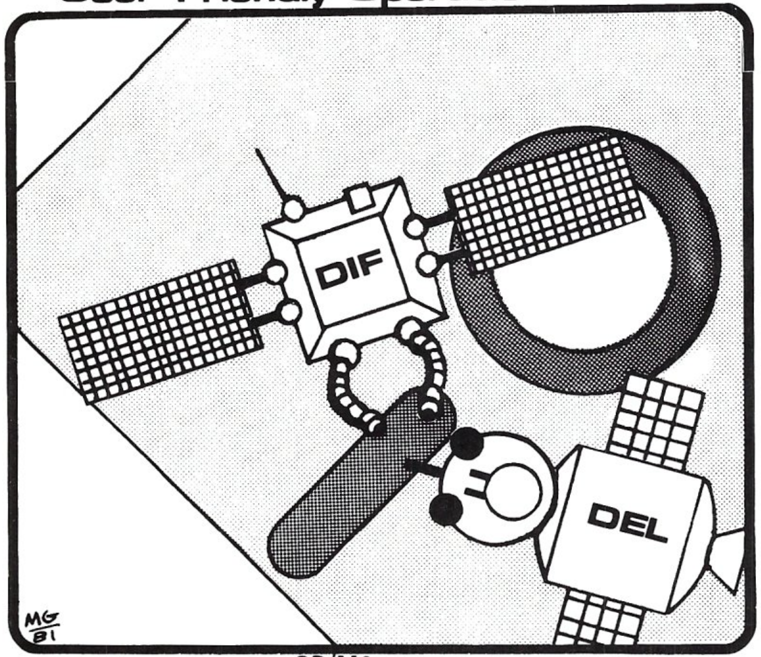
```
unitread(unitnum,directory,sizeof(directory),dirblk);
```

This statement uses two UCSD intrinsics, the "sizeof" function and the "unitread" procedure. The former returns the number of bytes in a given data structure; in our case it is in the number of bytes in the directory. The unitread procedure reads from unit (disk drive) "unitnum" into memory at the address of the directory structure for "sizeof(directory)" bytes starting at absolute disk block dirblk.

With this information in memory, we can search the directory for an entry of a given file. Suppose that we wanted to see if the file "sample.code" were in the directory. We could say

UFO's* Launched into File Space!

***User Friendly Operation**



CP/M® users:
whether you're working on text editing or debugging a program
DIF and DEL will save you lots of valuable time

DIF: The file difference detector
- will quickly show you the difference between any two files [source or object]
- will help automate keeping track of revisions made to your specifications, software releases, or any kinds of documents
- will present file changes as a difference report or as an annotated final copy

DEL: The file delete utility
- will help you be cautious and more productive
- will allow multiple files to be specified on a single command line, using all CP/M file reference forms
- will prompt you to confirm that you really want to delete the file before it is deleted
- will completely replace ERA

DIF and DEL — two User Friendly Operations designed to enhance your library of software tools

To place your order send $45.00 to:

DIGITAL CONSTRUCTS

Dept M, 130 E. Main St., Norristown, PA. 19401 (215) 279—5652

VISA, MasterCard, money order or personal check acceptable
Documentation $5.00 [refundable on purchase of full package]
Outside U.S.A. add $10.00 for shipping and handling.

DIF and DEL require CP/M 1.4 or 2.2 Diskette formats: single-density soft-sectored 8", Heath / Zenith 5¼" TPA at hex 0100 only.
CP/M is a trademark of Digital Research Dealer Inquiries Invited

Virtual Segment Procedures, continued...

```
numfiles := directory[0].dnumfiles; ( number of valid entries )
i := 0; ( will index into directory )
done := false; ( will become 'true' when we are done )
while (i < numfiles) and not done do begin
  i := i + 1;
  if (directory[i].dtid = 'sample.code') then done := true;
  end;
```

If we come out of this loop with "done" having a value of "true," then we have indeed found an entry for the required file; if not, the file was not present. If it was found, then the absolute address of the first block of the file on disk is simply "directory[i].dfirstblk". We have then found the first thing we needed, the absolute disk address of the code file. We now need to get the code length and relative disk address information from the code file's segment tables.

The format of the first block of a code file is as shown below, where the previous type definitions hold as before.

```
segtable : array[segrange] of segdesc;
```

If we wanted to find the necessary information about a particular code file, we could use the UCSD Pascal system intrinsic procedure unitread to read the segment table data from that file into the above data structure. Then, if we wanted the information about the tenth segment procedure in a code file, we could simply use the tenth element of that array. For example, to determine the length and block data for segment procedure number ten in a file called "sample.code" on unit five, one could do the following:

```
program find;
 var
   codefile : file;
   segtable : array[segrange] of segdesc; ( 'segdesc' defined as before )
   i : integer;
 begin
 unit := 5;
 ( read in segment table -- assume 'directory' read in as above first,
   and 'directory[i]' is data for file which is of interest to us )
 unitread(unit, segtable, sizeof(segtable), director[i].dfirstblk);
 length := segtable[10].codeleng;
 block := segtable[10].diskaddr; ( relative block number )
 block := block + directory[i].dfirstblk; ( absolute block number )
 end. ( find )
```

So, we finally have the entire ball of wax. We can read a UCSD disk directory to find out where on disk a file is stored; we can read and understand the information in the segment table of a code file; we can convert the relative block numbers in the code file segment tables to absolute block numbers; and we can use that information to invoke virtual segment procedures. To tie everything together, you will find below a single program which does it all. The procedure "virtinit" initializes an internal table of unit, length, and block information before the program really gets going. It does this by reading in the disk directory and calling "virtlink" for each virtual segment procedure which it will call later. The virtlink procedure looks in the directory for the code file requested and puts the required information in the internal table. The main program then invokes the virtual segments as a test. Following this program is a sample of one of the programs which defines a virtual segment procedure. Note that in both the program which calls the virtual segments and the program which defines one of them, the virtual segments are defined as the first executable code in the file. This is necessary, since the technique which I have shown requires that both segment procedures have the same segment numbers. ■

```
program vsegtest;

CONST
  MAXUNIT = 12;          (*MAXIMUM PHYSICAL UNIT # FOR UREAD*)
  MAXDIR = 77;           (*MAX NUMBER OF ENTRIES IN A DIRECTORY*)
  VIDLENG = 7;           (*NUMBER OF CHARS IN A VOLUME ID*)
  TIDLENG = 15;          (*NUMBER OF CHARS IN TITLE ID*)
  FBLKSIZE = 512;        (*STANDARD DISK BLOCK LENGTH*)
  DIRBLK = 2;            (*DISK ADDR OF DIRECTORY*)
  MAXSEG = 15;           (*MAX CODE SEGMENT NUMBER*)

TYPE

  DATEREC = PACKED RECORD
    MONTH: 0..12;              (*0 IMPLIES DATE NOT MEANINGFUL*)
    DAY: 0..31;                (*DAY OF MONTH*)
    YEAR: 0..100               (*100 IS TEMP DISK FLAG*)
    END (*DATEREC*) ;

  UNITNUM = 0..MAXUNIT;
  VID = STRING[VIDLENG];

  DIRRANGE = 0..MAXDIR;
  TID = STRING[TIDLENG];

  FILEKIND = (UNTYPEDFILE,XDSKFILE,CODEFILE,TEXTFILE,
    INFOFILE,DATAFILE,GRAFFILE,FOTOFILE,SECUREDIR);

  DIRENTRY = RECORD
    DFIRSTBLK: INTEGER;                  (*FIRST PHYSICAL DISK ADDR*)
    DLASTBLK: INTEGER;                   (*POINTS AT BLOCK FOLLOWING*)
    CASE DFKIND: FILEKIND OF
      SECUREDIR,
      UNTYPEDFILE:                       (*ONLY IN DIR[0]...VOLUME INFO*)
        (DVID: VID;                      (*NAME OF DISK VOLUME*)
         DEOVBLK: INTEGER;               (*LASTBLK OF VOLUME*)
         DNUMFILES: DIRRANGE;            (*NUM FILES IN DIR*)
         DLOADTIME: INTEGER;             (*TIME OF LAST ACCESS*)
         DLASTBOOT: DATEREC);            (*MOST RECENT DATE SETTING*)
     XDSKFILE,CODEFILE,TEXTFILE,INFOFILE,
     DATAFILE,GRAFFILE,FOTOFILE:
        (DTID: TID;                      (*TITLE OF FILE*)
         DLASTBYTE: 1..FBLKSIZE;         (*NUM BYTES IN LAST BLOCK*)
         DACCESS: DATEREC)               (*LAST MODIFICATION DATE*)
    END (*DIRENTRY*) ;

  SEGRANGE = 0..MAXSEG;
  SEGDESC = RECORD
    DISKADDR: INTEGER;     (* REL BLK IN CODE...ABS IN SYSCOM*)
    CODELENG: INTEGER      (*# BYTES TO READ IN*)
    END; (*SEGDESC*)

   vsegrange = 0..15; { virtual segment number }

   { segment table in syscomrec and in this program }
   segtabtype = array [segrange] of record
     codeunit : unitnum;
     codedesc : segdesc;
     end;

 var
   alias : record case boolean of { allow manual setup of ↑syscomrec }
     true : (i : integer);
     false: (p : ↑segtabtype);
     end;
   disk : file; { file to read directory and seg tables from }
   { global in which to store processed seg table for later use }
   vsegs : segtabtype;

 segment procedure virtual;
   begin
   writeln('I am the REAL segment procedure 10.');
   end; { virtual -- dummy }

 procedure dovirtual(vsegnum : vsegrange);
   begin
   if (vsegs[vsegnum].codeunit = 0)  then begin
     writeln('Attempt to execute unlinked virtual seg number ',vsegnum,'.');
     exit(dovirtual); { not linked }
     end;
   { load segment register 10 with segment data from i'th segment procedure }
   alias.p↑[10] := vsegs[vsegnum];
```

```
    { read in segment table }
    firstblk := directory[i].dfirstblk;
    unitread(unum,lsegtable,sizeof(lsegtable),firstblk);
    { enter data from segment 10 in found file into v seg table }
    vsegs[vsegnum].codeunit := unum;
    vsegs[vsegnum].codedesc.codeleng := lsegtable[10].codeleng;
    vsegs[vsegnum].codedesc.diskaddr := lsegtable[10].diskaddr + firstblk;
    writeln('Finished with association of file ',fname,'.');
    end; { virtlink }

  begin
  { initialize vseg table so that all units are zero (vseg undefined) }
  for i := 0 to 15 do vsegs[i].codeunit := 0;
  { set up pointer to syscomrec's seg table located by 'find' program }
  alias.i := 718 + 96; { 96 bytes in syscom rec before seg tables }
  { read directory into memory }
  unitread(unum,directory,sizeof(directory),dirblk,0);
  numfiles := directory[0].dnumfiles;
  writeln('Directory of unit ',unum,' read in.');
  { link file names with v seg numbers }
  virtlink('FAKE0.CODE',0);
  virtlink('FAKE1.CODE',1);
  virtlink('FAKE2.CODE',2);
  virtlink('FAKE3.CODE',3);
  virtlink('FAKE4.CODE',4);
  virtlink('FAKE5.CODE',5);
  virtlink('FAKE6.CODE',6);
  virtlink('FAKE7.CODE',7);
  end; { virtinit }

begin { main }
virtinit;
dovirtual(0);
dovirtual(1);
dovirtual(2);
dovirtual(3);
dovirtual(4);
dovirtual(5);
dovirtual(6);
dovirtual(7);
end.


program fake0;

  segment procedure fake0also;
    begin
    writeln('I think that I am virtual segment procedure 0.');
    end;

  begin { main }
  end.
  virtual;
  end; { dovirtual }

procedure virtinit;
  { initialize tables for calling v segment procedures }
  const
    unum = 5; { unit number where v segment procs are found }
  var
    i : integer; { temp var }
    directory : array [dirrange] of direntry; { holds disk directory }
    numfiles : dirrange; { number of files in disk dir }

  procedure virtlink(fname : string; vsegnum : vsegrange);
    var
      i : integer;
      done : boolean;
      firstblk : integer; { blk number of first blk of fake seg code file }
      lsegtable : array [segrange] of segdesc; { holds seg tbl from disk }
    begin
    { find file in directory }
    i := 0; done := false;
    while (i <= numfiles) and not done do begin
      i := i + 1;
      if (directory[i].dtid = fname) then done := true;
      end;
    if not done then begin
      writeln('unable to find file ',fname,'.');
      exit(virtlink);
      end;
    writeln('File ',fname,' found.');
```

Little-Ada* (Part II)

by Ralph E. Kenyon, Jr.

Editor's Note:
In this installment, the author discusses the operation of the run-time interpreter and compiler. Sample programs are included.
Parts three (run-time interpreter and source code) and four (compiler and object code) will be presented in upcoming issues of *Microsystems.* Refer to "Little-Ada (Part 1)," published in the Sept/Oct issue, for an introduction to Little-Ada.
The DOD does not recognize dialects of the Ada language, whether by supersetting or subsetting.

Little-Ada

Little-Ada version L/1 consists of a language design, a target machine design for intermediate code and a compiler to compile the language to the target machine code. The target machine is called the L-Machine.

Adding a system-dependent interpreter produces the specific L/1 implementation (in this case: Little-Ada version PolyMorphic L/1).

The Little-Ada design goal is to implement a subset of Ada, consistent with full Ada, which can support the basic Ada language structure and syntax. The purpose of the design is to produce a transportable compiler which can be implemented on various computers from the smallest to the largest, with an eye to future expansion to include more and more features of full Ada. The intended use of the L/1 version of Little-Ada was to support education in compiler principles and implementation methods in a classroom environment.

Little-Ada implements a minimal subset of Ada that preserves the block structuring and syntax forms in full Ada. Ada has been described as, in part, a block-structured language with strong typing that is Pascal-based. Little-Ada implements one aggregate (ARRAY), two objects (variable and constant), three control structures (subprogram, if-then-else, and loop-exit), relational and arithmetic operators and an assignment statement. Arithmetic operators include *, /, MOD, +, and -. All control structures in Ada and in Little-Ada conform to a general arrangement called a "comb." Examples include:

Program declaration **LOOP - EXIT** **IF - THEN - ELSE**

```
/-- PROCEDURE .. IS     /-- LOOP            /-- IF .. THEN
:    ..                 :    ..             :    ..
:-- BEGIN               :-- EXIT            :-- ELSIF .. THEN
:    ..                 :    ..             :    ..
\-- END;                \-- END LOOP;       :-- ELSE
                                            :    ..
                                            \-- ENDIF;
```

Ralph E. Kenyon, Jr., 1686 West Main Rd., Portsmouth, RI 02871.
*Ada is a trademark of the U.S. Department of Defense (Ada Joint Program Office).

Reserved Words

Little-Ada has eighteen reserved words:

ARRAY	ELSE	EXIT	LOOP	OF	THEN
BEGIN	ELSIF	INTEGER	MOD	PRAGMA	TYPE
CONSTANT	END	IS	NULL	PROCEDURE	WHEN

There are eighteen special symbols in Little-Ada.

```
+  -   >=  :=   *  /
<  >   <=  ..   =  :
(  )   /=  --   ;  _
```

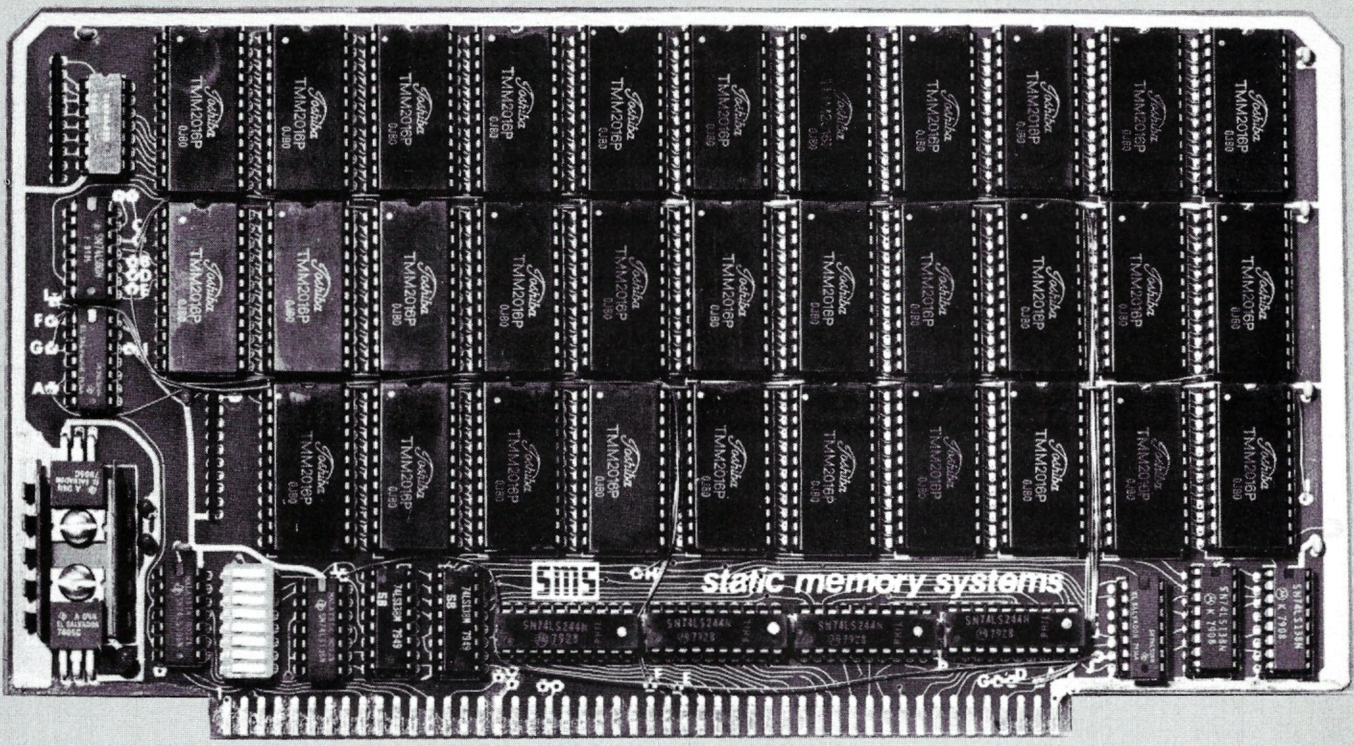
a. Six condition test symbols:
= /= < <= > >=
b. Four arithmetic operators:
+ - * / MOD
(MOD is included here for completeness.)
c. Assignment operator (becomes):
:= (as in NEW := OLD + 1;)
d. Object type indicator:
: (as in SIZE : INTEGER;
e. Range indicator (ellipsis read "to"):
.. (as in ARRAY (1..10) OF)
f. Comment (Terminated by end of line):
--
g. Statement terminator:
;
h. Parentheses:
( )
i. Underscore:
_ (ignored in numbers)

Declarations

In Little-Ada, all program units must be declared. Program units and sub programs are declared in exactly the same way. The following is an example.

```
PROCEDURE NAME IS
 -- declare part
BEGIN
 -- sequence of statements
END;
```

THE LAST MEMORY



64K STATIC RAM/EPROM BOARD

At last a 64K *STATIC* memory board for S100 systems. But it's not just a 64K static RAM board, *EPROM's* can also be intermixed with RAM making it the only memory board needed for S100 systems. That's why we call it **THE LAST MEMORY.**

• *64K DENSITY*
THE LAST MEMORY uses the new 2016 byte-wide 16K static RAM to achieve a board density twice that possible with old 2114 static memories.

• *2716 EPROM COMPATIBLE*
A separate board is no longer required for EPROM's containing monitors, bootstrap loaders, etc.
2716 EPROM's can be inserted into the board without modification.

• *SIMPLE ADDRESS DECODING*
Where memory is required, just plug a RAM or EPROM in the corresponding socket. Empty memory sockets occupy no memory space, providing compatibility with memory mapped I/O devices.

• *EXTENDED ADDRESSING*
THE LAST MEMORY includes the IEEE S100 extended addresses. These are fully decoded allowing expansion to a full 16 megabyte system memory.

• *FAST*
The standard board allows 4 MHz operation.

• *LOW POWER*
Only one memory IC is ever active in byte-wide memory systems. The result is far less power consumption than older 16K static memory boards.

• *LOW COST*
Its best feature is the price:

	Kit	**A&T**
RAM-less Board	99.99	139.99
16K RAM	249.99	289.99
32K RAM	389.99	429.99
48K RAM	519.99	559.99
64K RAM	639.99	679.99

SMS **static memory systems**
15 So. Van Buren Ave. Suite 209
Freeport, Illinois 61032 (815) 235-8713

VISA

Little Ada, continued...

The declare part must declare all objects not already declared that are subsequently referenced in the procedure. Little-Ada differs from full Ada in that Little-Ada (L/1) does not implement parameters, and locally declared variables hide global variables of the same name. Parameter passing can be achieved in Little-Ada in a manner that is simpler to implement, but does not provide for safer programming later. In Little-Ada, procedures are allowed access to all variables in the same scope (not hidden).

An ARRAY type may be declared with the following syntax:

```
TYPE NEW←TYPE IS ARRAY (LOW..HI) OF BASE←TYPE;
```

Examples:

```
TYPE VECTOR IS ARRAY (1..10) OF INTEGER;
TYPE MATRIX IS ARRAY (1..10) OF VECTOR;
```

Variables and constants are declared as follows:

```
VARIABLE←NAME : TYPE←NAME;
CONSTANT←NAME : CONSTANT INTEGER := VALUE;
```

Examples:

```
SIZE : INTEGER;
HIGHT : INTEGER := 10;
LINES←PER←PAGE : CONSTANT INTEGER := 66;
```

In Ada and Little-Ada, identifiers and numbers may contain single embedded underscores for readability. Examples:

```
LOW←IN←CH;  22←350
```

The underscore has no effect on numbers, but is part of identifiers. An identifier starts with a letter, but may contain digits and letters with single embedded underscores. Little-Ada only distinguishes the first ten characters in an identifier and limits numbers up to 32←767. Also, in Little-Ada only upper case may be used in identifiers and keywords. Comments may use both upper and lower case. Here Little-Ada differs from full Ada in that full Ada accepts lower case, but treats lower case as upper case in identifiers and reserved words.

Statements

Little-Ada implements only one of the Ada LOOP structures:

```
LOOP
--
EXIT;
END LOOP;
```

EXIT statements must be used to exit from a Little-Ada LOOP statement. The syntax is as follows:

```
EXIT;
-- or
EXIT WHEN condition;
```

Examples:

```
EXIT WHEN LOW←CHAR = 13; -- end of line
IF LOW←CHAR = 13 THEN EXIT; END IF;
-- (same effect)
```

Comparing loop structures in Basic, Fortran, Little-Ada and Pascal:

```
Little-Ada                 ;           Basic
---------------------------+----------------------------------
                           ;        10 REM
LOOP                       ;        20 REM   statements
  -- statements            ;        30 IF condition GOTO 60
EXIT WHEN CONDITION;       ;        40 REM   more statements
  -- statements            ;        50 GOTO 10
END LOOP;                  ;        60 REM
```

AUXILIARY PROCESSOR

- **Z-80 CPU**
- **S-100**
- **16K BYTES RAM**
- **UP TO 8 BYTES ROM**
- **8253 PROGRAMMABLE CLOCK**

The AUX-10 is a general purpose auxiliary processor which can either be used as a dedicated controller or as an additional processor in a multiprocessor system. The board incorporates 16K of RAM and up to 8K bytes of ROM to allow complex program execution. The board can either execute programs directly from the on-board ROM or from programs loaded from the main processor. The main processor communicates with the slave processor through a common memory on the slave processor board. Commands and data are transferred in this memory space. In addition the card has an 8253 programmable clock which not only the auxiliary processor can use but also the main processor. The board's internal bus is brought off board to allow dedicated controller applications. This allows the slave processor to be used as an intelligent controller with external peripherals. The board operates at 4 MHz with no wait states.

DIGITAL SYNTHESIZER

- **32 CHANNELS**
- **AMPLITUDE AND FREQUENCY CONTROL**
- **FREQUENCY MODULATION**
- **UP TO 16 WAVEFORM STORAGE**
- **PROGRAMMABLE TIMBRE WAVEFORMS**

Casheab has designed and developed a 32 channel digital sound synthesizer for the S-100 bus. The synthesizer consists of two cards: a synthesizer card (SYN-10) and a controller card (CTR-10). The S-100 host processor programs the waveforms (1024 by 12 bits) into the synthesizer and up to 16 waveforms can be stored. Any of the channels can use any of the waveforms. In addition attack, steady state and decay envelopes can be implemented by the host processor controlling each channel's amplitude. The synthesizer also incorporates frequency modulation which can be used for vibrato.

Software on a CP/M* compatible floppy disk is provided free with the purchase of the synthesizer.

*CP/M is a trademark of Digital Research

CASHEAB
5737 AVENIDA SANCHEZ
SAN DIEGO, CA 92124
277-2547

SYN-10/16 & CTR-10	1245.00
MANUAL	5.00
DEMO CASSETTE	3.00
AUX-10	345.00
MANUAL	5.00

```
           Fortran             :            Pascal
-------------------------------+---------------------------
                               :   LABEL 10,20;
                               :
10         CONTINUE            :   10 NIL;
c          statements          :      { statements; }
           IF condition GOTO 20:     IF condition THEN
c          more statements     :         GOTO DONE;
           GOTO 10             :      { more statements; }
20         CONTINUE            :     GOTO 10;
                               :  20 NIL;
```

```
           Other Pascal loops                    : Little-Ada
-------------------------------------------------+---------------
           -          : REPEAT                   : LOOP
           -          :    BEGIN                 :   --
           -          :    { statements; }       :   -- statements
           -          :    END;                  :   --
WHILE NOT condition   : UNTIL condition;         : EXIT WHEN cond;
  BEGIN               :           -              :   --
    { statements; }   :           -              :   -- statements
  END;                :           -              :   --
END;                  :           -              : END LOOP;
```

Full Ada has a FOR K IN range LOOP .. END LOOP; (where "range" is from LOWER←BOUND to UPPER←BOUND), which can be accomplished in Little-Ada as follows:

```
I : INTEGER;

I := LOWER←BOUND;
LOOP
EXIT WHEN I>UPPER←BOUND;
  -- statements
  I := I + 1;
END LOOP;
```

Corresponding structures are:

```
Basic              : Fortran          : Pascal
-------------------+------------------+-------------------
10 FOR I=1 TO N    :   DO 20 I=1,N    : FOR I := 1 TO N DO
20 REM statements  : C  statements    :   BEGIN
30 NEXT I          : 20 CONTINUE      :     { statements; }
                   :                  :   END;
                   :                  : END;
```

Conditional control structure in Little-Ada and in full Ada uses the IF statement. The syntax is:

```
IF condition THEN statements;
ELSIF condition THEN statements;
ELSE statements;
END IF;
```

ELSEIF and ELSE are optional. Examples:

```
IF SCORE > 95                  IF DOOR = OPEN
  THEN MARK := A;                THEN CLOSE;
ELSIF SCORE > 85               END IF;
  THEN MARK := B;
ELSIF SCORE > 75
  THEN MARK := C;              IF SEX = MALE
ELSIF SCORE > 65               THEN
  THEN MARK := D;                COLOR := BLUE;
ELSE                           ELSE
  MARK := F;                     COLOR := PINK;
END IF;                        END IF;
```

The primary statement is the assignment statement.

```
RESULT := EXPRESSION;
```

Numerical computations include +, -, *,/ and MOD, and are all 16-bit signed integers. Multiplicative operators *, /, and MOD take presidence over additive operators + and -. Multiplicative operations should be parenthetically grouped to prevent ambiguity. A * B / D is ambiguous and should be written (A * B) D, or A * (B / D).

L/1 Machine

The L/1 version of the L - Machine is a 16-bit stack-oriented processor with a single 8-bit I/O port and a simple instruction set consisting of six primary and eighteen secondary instructions.

L/1 Compiler

The compiler implements only one data type—16-bit integers. INTEGER is a built-in type. One built-in variable, LOW←CHAR, is used to load the I/O port. Two build-in procedures, LOW←IN←CHAR and LOW←OUT←CH input and output from LOW←CHAR through the I/O port. While Ada allows use of upper and lower case, Little-Ada only recognizes upper case in program identifiers, while comments may include lower case.

Additional limitations of the compiler include a limit of 2000 bytes of code and fifty symbols. Because of the recursive descent technique used by the compiler, the fifty symbols mean a maximum of fifty symbols at a given moment during compilation. Space for local names within a procedure is reclaimed when that procedure has completed compiling. This allows more than fifty total symbols, provided the current stage is never more than fifty. Procedures may be nested to a maximum of fifteen levels (LOW←IN←CHAR and LOW←OUT←CH are one level below the main program). Input lines may be a maximum of 81 characters (including the CRLF). Each object declaration is limited to a maximum of seventeen identifiers per statement.

There are some more subtle limitations to the compiler. It does not make any run-time checks of array limits. The compiler will also not detect some kinds of syntax errors. Initializing a CONSTANT or a variable will not detect an error if an identifier is used instead of a number. Assignments of an array will not assign the entire array; only the first element is assigned.

INFOSOFT Has a Better Way

C

FULLY INTEGRATED "C" AND ASSEMBLER DEVELOPMENT TOOL KIT

FEATURING INFOSOFT'S I/SAL T.M.

InfoSoft SYSTEMS INCORPORATED
25 SYLVAN ROAD SOUTH
WESTPORT, CONN 06880
(203) 226-8937

Sample Programs

The following example is an illustration of a simple program to output a carriage return.

```
PROCEDURE CR IS
 --
BEGIN -- CR
 LOW←CHAR := 13;
 LOW←OUT←CH; -- outupt LOW←CHAR
END; -- CR
```

The following example illustrates the use of this program as a subprogram in another program called HI. Subprograms are invoked by mentioning their names.

```
PROCEDURE HI IS
  PROCEDURE CR IS
  BEGIN -- CR
    LOW←CHAR := 13;
    LOW←OUT←CH; -- outupt LOW←CHAR
  END; -- CR
BEGIN -- HI
  LOW←CHAR := 72; -- H
  LOW←OUT←CH; -- output "H"
  LOW←CHAR := 105; -- i
  LOW←OUT←CH; -- output "i"
    CR; -- output a CR too
  END; -- HI
```

The following is a more significant program. This program illustrates pseudo parameter passing as well as programming style. It prints out a list in columns of numbers from zero to fourteen which includes the number, the powers of two, the squares of the number, the sum of the numbers, etc.

```
PROCEDURE EXAMPLE IS
  N : INTEGER; -- N is the global IN parameter
  END←OF←LINE : CONSTANT INTEGER := 13;
  W,X,Y,Z,J : INTEGER;
  --
  PROCEDURE PRINT←N IS
    TYPE NUMBER IS ARRAY (1..6) OF INTEGER;
    -- ASCII constants
    ZERO : CONSTANT INTEGER := 48;
    BLANK : CONSTANT INTEGER := 32;
    MINUS : CONSTANT INTEGER := 45;
    -- variables
    DIGIT : NUMBER;
    X,INDEX : INTEGER; -- X is work copy
  BEGIN -- PRINT←N
    -- Copy global parameter into local IN parameter X
    X := N;
    -- first, check sign
    IF X < 0
    THEN -- sign goes in the '6th' digit position
      DIGIT (6) := MINUS - ZERO; -- we add ZERO on output
      X := -1 * X;  -- MOD only works on positive numbes
    ELSE
      DIGIT (6) := BLANK - ZERO;
    END IF;
    --
    INDEX := 0;
    LOOP -- now we put each digit in its array slot
      INDEX := INDEX + 1;
      DIGIT (INDEX) := X MOD 10;
      X := X / 10;
    EXIT WHEN INDEX = 5;
    END LOOP;
    --
    LOOP -- now we blank leading zeros and move sign
    EXIT WHEN INDEX = 1;
      IF DIGIT (INDEX) = 0
      THEN -- move sign
         DIGIT (INDEX) := DIGIT (INDEX + 1);
         -- and blank previous position
         DIGIT (INDEX + 1) := BLANK - ZERO;
      ELSE
    EXIT; -- if the digit wasn't zero we want to quit
      END IF;
      INDEX := INDEX - 1;
    END LOOP;
    --
    LOW←CHAR := BLANK;
    LOW←OUT←CH; -- Two leading blanks
    LOW←OUT←CH; -- on each number
   INDEX := 6;
    LOOP -- now we print out all six digits
    EXIT WHEN INDEX = 0;
      LOW←CHAR := DIGIT (INDEX) + ZERO;
      LOW←OUT←CH;
      INDEX := INDEX - 1;
    END LOOP;
  END; -- PRINT←N


    --
    BEGIN -- EXAMPLE
     W := 0;
     J := 0;
     Y := 0;
     Z := 1;
     LOOP
       N := J;            -- INDEX
       PRINT←N;
       N := Z;            -- 2↑INDEX
       PRINT←N;
       N := J * J;        -- INDEX↑2
       PRINT←N;
       N := Y;            -- SUM J=0 TO N
       PRINT←N;
       N := W;            -- SUM J↑2 TO N
       PRINT←N;
       N := Z - W;        -- 2↑N - SUM N↑2
       PRINT←N;
       LOW←CHAR := END←OF←LINE;
       LOW←OUT←CH;
     EXIT WHEN J = 14;
       Z := Z * 2;
       J := J + 1;
       W := W + J * J;
       Y := Y + J;
     END LOOP;
   END; -- EXAMPLE
```

Program EXAMPLE Output

```
  0        1       0       0       0       1
  1        2       1       1       1       1
  2        4       4       3       5      -1
  3        8       9       6      14      -6
  4       16      16      10      30     -14
  5       32      25      15      55     -23
  6       64      36      21      91     -27
  7      128      49      28     140     -12
  8      256      64      36     204      52
  9      512      81      45     285     227
 10     1024     100      55     385     639
 11     2048     121      66     506    1542
 12     4096     144      78     650    3446
 13     8192     169      91     819    7373
 14    16384     196     105    1015   15369
```

Bower-Stewart & Associates SOFTWARE AND HARDWARE DESIGN

$GOLD DISK$ CP/M® Compatible Z-80 Software

Available for all 8-5" SS-SD IBM format systems including TRS-80®, Northstar, SD Systems. Also available on 5" double density Superbrain.®

Un-can your canned software!

$175. ppd

Z-80 Disassembler Feel couped up with your canned software? Our Z-80 Disassembler recreates assembly language source files from absolute code enabling users to easily tailor programs to meet their specific needs. The Preconditioner works with the Disassembler to decode ASCII.

Great looking letters & reports!

$75. ppd

E-Z Text A unique word processor organized around user-created text files, embellished with simple control commands, which supports such 'BIG GUYS' features as Automatic Footnoting, Table Spacing, Heading, Paging, Left & Right Margins, Proportional Spacing and MORE, at a 'LITTLE GUYS' price tag.

Credit cards: Immediate service, free 24 hr. phone – we will credit invoice. Checks, M.O.'s: Ten workday hold. CA. res: Add tax.

VISA

MasterCard

State system & controller. Allow time for surface mail. Trademarks: Digital Research, Radio Shack, Intertec.

POST OFFICE BOX 1389   HAWTHORNE, CALIFORNIA 90250   213 / 676-5055

```
                            BLOCK

 :
 : ..............................<;>...[type]..........
 :/                                                    \
 :                                                     :
 :\...<TYPE>..[identifier]...<IS>...........           :
 :                                          \          :
 :                                          :          :
 :   .................<ARRAY>............../           :
 :  /                                                  :
 : :                                                   :
 :  \..<(>...[number]...<..>...[number]...<)>...<OF>.../
 :
 : .........................................<;>.....
 :/                                                 \
 :                                                  :
 :\...[identifier]..<:>......................[type]..../:
 : /             \    \                   /     \    :
 : :             :    :                   :     :    :
 :  \......<,>..../     \..<CONSTANT>../         :    :
 :                ................................../   :
 :               /                                    :
 :               \.....<:=>.......[number]................/
 : ..............................................
 :/                                              \
 :                                               :
 :\.....<PROCEDURE>..[identifier]...<IS>...[block]....../
 :
 :
  \...<BEGIN>...[sequence of statements]....<END>...<;>...
```

```
                              NAME
.........................[identifier]......................
  \                                                      /
   :                                                    :
    \...[name]........<(>.....[expression]....<)>..../
```

```
                    NUMBER
           ..........[digit].............
             /                \
            :                  :
            :\...<underscore>.../:
            :                  :
             \................./
```

```
                     PROGRAM
......<PROCEDURE>....[identifier]....<IS>....[block].....
```

```
                    SEQUENCE OF STATEMENTS

                 \...........          ............./
                             \        /
                             :      :
 ........................../.......\...<;>................
/                                                          \
:                                                          :
:\......[variable name].....<:=>......[expression].........../:
:                                                          :
:\.........[subprocedure identifier]........................./:
:                                                          :
:\....<LOOP>...[sequence of statements]......<END>..<LOOP>.../:
:                                                          :
:\...<EXIT>..........<WHEN>....[condition]................../:
:                   \                                      :
:                    :                                     :
:                     \.................................../:
:                                                          :
:\.<IF>....[condition]..<THEN>..[sequence of statements]..   :
:       /                                              \   :
:      :                                               :   :
:       \....<ELSIF>................................../:   :
:                    /                                 :   :
:                   :                                  :   :
:                    \..<ELSE>..[sequence of statements]...  :   :
:                                                      \:   :
:                                                      :   :
:    ................................................../   :
:   /                                                      :
:  :                                                       :
:   \....<END>..<IF>......................................./:
:                                                          :
\.................<NULL>................................../
```

```
                     IDENTIFIER

         ..............[letter]...
                  /              \..............
                 :  ...[digit].../  \
                 :/                  :
                 :\...<underscore>.../:
                 :                   :
                  \................./
```

```
                              CONDITION

....[expression]...........<=>...........[expression]....
                  \                    /
                   :                  :
                   :\..</=>.../:
                   :\...<<>.../:
                   :\..<<=>.../:
                   :\...<>>.../:
                    \...<>=>../
```

```
                      EXPRESSION

 .............................[term]......
     \                       /        \
      :         ....<+>..../:          :
       \...../             :          :
       /        \....<->..../           :
      :                                :
       \............................./
```

```
                         TERM

 ..................[factor]..................
               /               \
              :                 :
              :\.....<*>....../:
              :\....</>......./:
               \....<MOD>..../
```

```
                          FACTOR

.......................[number]..........................
      \                                               /
       :                                             :
       :\................[name]................/:
       :                                             :
        \....<(>....[expression]....<)>....../
```

Little-Ada (L/1) Syntax Diagrams

Block	Identifier
Name	Condition
Number	Expression
Program	Term
Sequence of Statements	Factor

Table V: *Little-Ada. L/1 Compiler Error Listing.*

Error	Interpretation	Error	Interpretation
001	Program too big! Code overflow!	031	Missing "LOOP" on END statement!
002	Compiler error! Patch not a branch!	032	No EXIT from LOOP!
003	Undefined base type!	033	EXIT statement begin error!
004	Inappropriate base type!	035	Undefined identifier!
005	Too many names! Symbol table overflow!	036	Not variable nor subprogram!
006	Number too large!	037	EXIT not allowed here!
007	Illegal character > "Z"!	038	Missing semicolon at statement end!
008	Illegal character > "9" and < "A"!	039	Compiler error! TYPE token lost!
009	Illegal character < "0"!	040	Missing type name!
010	Number too large!	041	Missing IS!
011	Undefined identifier!	042	Missing "ARRAY" in type declaration!
012	Array of invalid type!	043	Missing left parenthesis!
013	Missing right parenthesis!	044	Missing lower bound!
014	Illegal use of identifier!	045	Missing ".."!
015	Missing right parenthesis!	046	Missing upper bound!
016	Illegal factor beginning!	047	Missing right parenthesis!
017	Illegal relational operator!	048	Missing OF!
018	Illegal subscripting!	049	Missing base type name!
019	Missing right parenthesis!	050	Too many identifiers in declaration!
020	Missing ":="!	051	Missing semicolon!
021	IF statement begin error!	052	Missing semicolon after unit body!
022	Compiler error! THEN token lost!	053	Level too deep!
023	Missing THEN!	054	Missing BEGIN on unprefixed block!
024	Missing END!	055	Missing END on unprefixed block!
025	Compiler error! IF token lost!	065	Subprogram begin error!
026	LOOP statement begin error!	066	Missing subprogram name!
030	Missing "END" at end of LOOP statement!	067	Missing IS in procedure declaration!
		068	Semicolon expected after last "END"!

Little-Ada.L1 Test Program

This is an example of a Little-Ada.L1 test program. The output here is exactly as it appeared running the L/1 Interpreter on the block of code developed by Dr. Mathis. The block of code is the transportable compiler. In the listing, the code index is followed by a copy of the input line of text. Errors will show up in the listing if any exist (none shown here). See Table V for a complete list of errors generated by the Little-Ada compiler. The test program listing is followed by an ASCII block of the output code (which is the L/1 object code for this program).

CODE
INDEX--SOURCE CODE--

```
0006 PROCEDURE TEST IS
0006 TYPE NUMBER IS ARRAY(1..5) OF INTEGER;
0006 VAL:NUMBER; I,X,Y:INTEGER;
0014  -- A program to print the number 00256
0014 BEGIN
0016   X:=256;
0022   -- This routine puts digits into array value
0022   I:=1; LOOP EXIT WHEN I>5;
0038   Y:=X/10; VAL(I):=X-Y*10; X:=Y;
0078   I:=I+1; END LOOP;
0090   -- This routine prints out the digits top first
0090   I:=5; LOOP EXIT WHEN I=0;
0106   LOW←CHAR:=VAL(I)+48; LOW←OUT←CH;
0127   I:=I-1; END LOOP;
0139  --
0139 END;
**
0006 908C 918C A58F A18F A18F A18F 0010 C000 09B1 008E C000
0AA1 8EC0 000A 8DA5 8A87 4026 005A C000 08C0 0009 8DAA 848E
C000 03C0 000A 8DA1 8281 C000 098D C000 088D AA83 828E C000
09C0 0008 8D8E C000 0AC0 000A 8DA1 818E 001B C000 0AA5 8EC0
000A 8DA0 8887 406A 008B C100 00C0 0003 C000 0A8D A182 818D
B030 818E D100 04C0 000A C000 0A8D A182 8E00 5F8C 0000 0000
```

The above block of code was "hand disassembled" to produce the L/1 assembly source listing shown below. The comments show the relationship of the Little-Ada.L1 test program to the generated code.

```
ADDR CODE     LABEL   OPCODE          COMMENT
                      ORG 0

0000 0006             br Main

                ;This is a subroutine for the built
                ;in routine to input a character

0002 90       IN←CH   inb             ;LOW←IN←CH
0003 8C               ret

              ;This is a subroutine for the built
              ;in routine to output a character

0004 91       OUT←CH  outb            ;LOW←OUT←CH
0005 8C               ret

              ;Code generated for TEST.AD starts
              ;here.  Initially, space is reserved
              ;on the stack for the declared
              ;variables and constants. Each is
              ;done separately, as it is
              ;processed by the parser.

0006 A5       Main    lic 5           ;VAL
              VAL     EQU     3
0007 8F               inc
0008 A1               lic 1           ;Y
              Y       EQU     8
0009 8F               inc
000A A1               lic 1           ;X
              X       EQU     9
000B 8F               inc
000C A1               lic 1           ;I
              I       EQU     0AH
000D 8F               inc

              ;After all declarations are processed,
              ;the program branches past any
              ;subprocedures to the beginning of
              ;the main part. Here, there are no
              ;subprocedures so the branch is
              ;to the next address.

000E 0010             br Start

              ;The assignment statement X := 256;

0010 C0 0009  Start   lad 0,X
0013 B1 00            lic 256
0015 8E               sto             ;X:=256

              ;The assignment statement I := 1;

0016 C0 000A          lad 0,I
0019 A1               lic 1
001A 8E               sto             ;I:=1

              ;The LOOP statement does nothing
              ;except establish the label LOOP1
              ;for the backward branch at the
              ;corresponding END LOOP;

              ;The EXIT WHEN condtion statement
              ;caused the condition code
              ;generation first and the
              ;conditonal branch for exit.
```

```
ADDR CODE        LABEL   OPCODE          COMMENT

001B C0 000A     LOOP1   lad 0,I
001E 8D                  rav
001F A5                  lic 5
0020 8A                  setst           ;WHEN I>5
0021 87                  not
0022 4026                bnz BLOCK1
0024 005A                br  BLOCK2      ;EXIT

0026 C0 0008     BLOCK1  lad 0,Y
0029 C0 0009             lad 0,X
002C 8D                  rav
002D AA                  lic 10
002E 84                  div
002F 8E                  sto             ;Y:=X/10
0030 C0 0003             lad 0,VAL
0033 C0 000A             lad 0,I
0036 8D                  rav
0037 A1                  lic 1
0038 82                  sub
0039 81                  add      ;VAL(I)=VAL+(I-1)
003A C0 0009             lad 0,X
003D 8D                  rav
003E C0 0008             lad 0,Y
0041 8D                  rav
0042 AA                  lic 10
0043 83                  mul             ;10*Y
0044 82                  sub             ;-X
0045 8E                  sto             ;VAL(I)
0046 C0 0009             lad 0,X
0049 C0 0008             lad 0,Y
004C 8D                  rav
004D 8E                  sto             ;X:=Y
004E C0 000A             lad 0,I
0051 C0 000A             lad 0,I
0054 8D                  rav
0055 A1                  lic 1
0056 81                  add
0057 8E                  sto             ;I:=I+1
0058 001B                br LOOP1
005A C0 000A     BLOCK2  lad 0,I
005D A5                  lic 5
005E 8E                  sto             ;I:=5
005F C0 000A     LOOP2   lad 0,I
0062 8D                  rav
0063 A0                  lic 0
0064 88                  sete            ;WHEN I=0
0065 87                  not
0066 406A                bnz BLOCK3
0068 008B                br End          ;EXIT

006A C1 0000     BLOCK3  lad 1,LOW←CHAR
006D C0 0003             lad 0,VAL
0070 C0 000A             lad 0,I
0073 8D                  rav
0074 A1                  lic 1
0075 82                  sub
0076 81                  add
0077 8D                  rav      ;VAL(I)=VAL+(I-1)
0078 B030                lic '0'
007A 81                  add
007B 8E                  sto      ;LOW←CHAR:=VAL(I)+'0'
007C D1 0004             call 1,OUT←CH
007F C0 000A             lad 0,I
0082 C0 000A             lad 0,I
0085 8D                  rav
0086 A1                  lic 1
0087 82                  sub
0088 8E                  sto             ;I:=I-1
0089 005F                br LOOP2

008B 8C          End     ret

                         END
```

Summary

Emulator programs can be written for machines that do not exist. These non-exsistant machines are useful in the design of hardware machines, as well as in the design of high level languages. One such emulator on the 8080A in the PolyMorphic 8813 is being used in the development of a compiler for a subset of the new DOD language Ada called "Little-Ada." Ada is a strongly typed block structured language being designed for use in DOD embedded computer systems. The emulated machine is called the L - Machine and has been described in detail. This emulation also shows how a stack-oriented machine of radically different characteristics than the 8080A can be implemented on an 8080A. ■

New Software for

NorthStar

Users

EXPENSE PROFILE

- ☐ Find out where your money goes ! Summarize expenses by use and user.
- ☐ Simplify tax preparation. Calculates tax deductibles for each family member separately. Easy JOINT vs SEPARATE RETURN comparisons.
- ☐ Menu Driven. Interactive input. Add new spending categories and spenders any time. Find records by searching for any part of line, eg: "car".
- ☐ Correct mistakes anytime. Change wrong data only, not the whole line.
- ☐ Records stored on disc. Limited only by disc capacity. Output to TV or printer.

$29.95 ppd

DYNAMIC BUDGET

- ☐ Wonder if you can afford that new printer or boat or ?? Find out with DYNAMIC BUDGET !
- ☐ Easy input. Periodic items like salary and rent are entered only once, even if periods are in weeks, quarters, etc.
- ☐ Includes INFLATION and CALENDAR.
- ☐ Output to printer or TV. Monthly listings of expenses and income. Calculates running balance.
- ☐ Interactive. Change data anytime and get new results immediately.
- ☐ Data stored on disc. Makes keeping budget up to date easy.

$29.95 ppd

The Software Connection
10703 Meadowhill Rd.
Silver Spring, MD 20901

☐ Send me EXPENSE PROFILE
☐ Send me DYNAMIC BUDGET

I am enclosing $ ________

Name ____________________
Address ____________________
City,State ____________________
Zip ____________________

The Tarbell Double Density Disk Interface

by Fred Greeb

You want to use a DMA controller with dynamic memory? Good luck! That was the typical reaction when I said that I was considering ordering the Tarbell Double Density disk controller board. For those of you who don't know, a DMA controller performs data transfer directly to and from the computer memory, without any processor intervention required. It essentially turns off the main system processor, and the processor timing, and generates its own timing signals for data transfer. Dynamic memory boards, such as the SD Sales Expandoram which I have in my system, tend to be very fussy and require that the timing signals be just right for proper operation. That's why I got the reaction that I did. I didn't pay any attention to these comments, and went right ahead and ordered the controller. This article describes my experience in getting the board running in my system.

Selection of the Controller

There were several reasons why I selected the Tarbell controller. I plan to expand my computer in the future to include multi-user software, and wanted a disk controller which would not tie up the system during disk data transfer. If the controller requires the processor to handle the data, normal terminal input cannot be accomplished during disk accesses. This can be very annoying in a multi-user environment.

To meet this requirement, I needed either a DMA controller, or a controller with an on-board buffer for temporary storage of the disk data. The controllers with on-board memory buffers that I looked at all required that the controller memory occupy part of the main memory space, usually somewhere in the top 16K of the normal 64K addressing range of an 8-bit microprocessor. Since I wanted to retain the capability to expand the memory beyond 64K, with extended addressing rather than bank select, this was not an acceptable option.

The Tarbell controller implements full 24-bit extended addressing during DMA data transfers, utilizing the additional address lines defined in the IEEE-696/S-100 bus specification. This feature would be very useful if and when I expand the system to either multi-user software or 16-bit processor.

Finally, I have been running the Tarbell single density disk controller for over three years, and have not had any problems with it. The single density controller has proven to be a very reliable unit. This fact gave me confidence in Tarbell's capability to design and manufacture a double density interface. Since I already had two 8" disk drives, the fact that the controller does not contain the option to use 5 1/4" mini drives was of no concern.

The Controller Arrives

I ordered the controller board from one of the mail order computer suppliers which regularly advertises discount prices in most of the computer magazines. I ordered an assembled and tested unit, since I was told that Tarbell no longer sells the controller in kit form. The unit I received was marked Revision G. Hopefully, I thought, with that many revisions, all of the bugs will have been removed from the design.

The board is rather densely packed with about fifty integrated circuits, all socketed, and numerous discrete components to handle the disk to computer interface. Most of the jumper options that were available on the Tarbell single density controller have been removed. The double density unit is defined as Shugart-compatible only, and does not contain jumper options to configure it for other drives. Shugart-compatible does not mean that you have to use Shugart drives. It only requires that your drives utilize the same interface signals on the same connector pins as the Shugart drives. The majority of the drives currently available meet this requirement.

The documentation which arrived with the controller consisted of a manual and a disk. The manual is adequate, but not overly detailed in the technical description section. There is a very good section on getting the controller running, especially if you have been running the single density controller as I had been. The manual contains over two pages covering "what to do if it doesn't work," and listing symptoms and possible cures. It also contains a list of compatible processor boards, memory boards,

Fred Greeb, 1915 S. Cape Way, Denver, CO 80227.

S-100 Modem

An Atlanta bulletin board system uses a Hayes S-100 modem around the clock. Since March 1979, it has logged over 21,500 calls and been down a mere 10 minutes. For performance like this, depend on the Hayes Micromodem 100.™ Features include automatic dialing/answering, 45 to 300 baud operation, a built-in serial interface and direct connection to any modular phone jack.

The Micromodem 100 — and Micromodem II™ for Apple II* computers — are now available nationwide. Call or write for the name of your nearest dealer.

Hayes™

Hayes Microcomputer Products Inc.
5835 Peachtree Corners East, Norcross, GA 30092 (404) 449-8791

Micromodem 100 & Micromodem II are trademarks of Hayes Microcomputer Products Inc. *TM Apple Computer Inc.

How Much is Your Sanity Worth?

	List Price	Our Price
WordStar Version 3	495	425
SpellStar (requires WordStar Version 3)	250	225
MailMerge Version 3 (requires WordStar Version 3)	125	110
All Three Above	870	725
Condor DBMS Level II Relational Data Base	995	850
C. Itoh Starwriter I 25 cps Daisywheel Printer	1895	1650
C. Itoh Starwriter II 45 cps Daisywheel Printer	2495	2250
Televideo 950-C Terminal With Detachable Keyboard	1195	1050

All orders should be prepaid
Add 5% shipping charges for hardware.
NJ residents add 5% sales tax.

Information Technologies
38-67 Taylor Road
Fair Lawn, NJ 07410
201-796-3140
Technical Hotline: 617-755-8691

AUXILIARY PROCESSOR

- **Z-80 CPU**
- **S-100**
- **16K BYTES RAM**
- **UP TO 8 BYTES ROM**
- **8253 PROGRAMMABLE CLOCK**

The AUX-10 is a general purpose auxiliary processor which can either be used as a dedicated controller or as an additional processor in a multiprocessor system. The board incorporates 16K of RAM and up to 8K bytes of ROM to allow complex program execution. The board can either execute programs directly from the on-board ROM or from programs loaded from the main processor. The main processor communicates with the slave processor through a common memory on the slave processor board. Commands and data are transferred in this memory space. In addition the card has an 8253 programmable clock which not only the auxiliary processor can use but also the main processor. The board's internal bus is brought off board to allow dedicated controller applications. This allows the slave processor to be used as an intelligent controller with external peripherals. The board operates at 4 MHz with no wait states.

DIGITAL SYNTHESIZER

- **32 CHANNELS**
- **AMPLITUDE AND FREQUENCY CONTROL**
- **FREQUENCY MODULATION**
- **UP TO 16 WAVEFORM STORAGE**
- **PROGRAMMABLE TIMBRE WAVEFORMS**

Casheab has designed and developed a 32 channel digital sound synthesizer for the S-100 bus. The synthesizer consists of two cards: a synthesizer card (SYN-10) and a controller card (CTR-10). The S-100 host processor programs the waveforms (1024 by 12 bits) into the synthesizer. Either 4 waveforms (SYN-10/4) or 16 waveforms (SYN-10/16) can be stored. Any of the channels can use any of the waveforms. In addition attack, steady state and decay envelopes can be implemented by the host processor controlling each channel's amplitude. The synthesizer also incorporates frequency modulation which can be used for vibrato or FM synthesis.

Software on a CP/M* compatible floppy disk is provided free with the purchase of the synthesizer.

*CP/M is a trademark of Digital Research

CASHEAB
5737 AVENIDA SANCHEZ
SAN DIEGO, CA 92124
(714) 277-2547

SYN-10/4 & CTR-10	$1095.00
SYN-10/16 & CTR-10	1245.00
MANUAL	5.00
DEMO CASSETTE	3.00
AUX-10	345.00
MANUAL	5.00

and I/O boards which have been tested with the controller. The last half of the manual contains specification sheets for all of the integrated circuits used in the controller, including the 8257 DMA controller chip, and the 1793 floppy disk controller chip. A complete schematic of the controller takes up the last two pages of the manual.

The disk included with the interface is labelled "Public Domain Disk #2," with a copyright notice which seems to conflict with the public domain statement. The content of the disk is extremely helpful. Included on the disk are all the I/O drivers necessary to make the double density controller operate with CP/M, and several disk formatting and testing programs. If all manufacturers would supply programs required to make their hardware operate, in machine readable form, life would be easier for the microcomputer enthusiast.

Bringing Up the Controller

Bringing up the controller was very easy, since I already had a working disk system. I merely copied the BIOS file from the Public Domain disk, edited it to match the requirements of my terminal, and assembled the edited file. After merging the assembled file with my CP/M operating system, I had a disk ready to run with the new controller.

I installed the new controller board, powered up the computer, and tried to boot the newly created disk. It didn't work! Not to be discouraged, I tried booting the bootstrap ROM contained on the controller board. This time everything worked perfectly. The disk loaded, and everything appeared to be operating normally. The only problem was with the bootstrap ROM. This is apparently a common problem, and is dependent on the processor board (and motherboard) which you are using. The manual discusses this problem, and gives the probable cures. In my case, changing a capacitor from 100 pf to 370 pf solved the problem. The controller was working fine, including the bootstrap operation.

Response from Tarbell Electronics:

We have never claimed to be completely compatible with the proposed IEEE-696 standard, because we found it difficult to determine how close we were to the standard. We have, however, made it our policy to design our boards as close to the standard as possible without compromising features.

Revision G of our board was designed specifically to work with Ithaca InterSystems 64K dynamic memory board, and will still work with the Measurement System 64K dynamic memory board. We have too little experience with other dynamic boards to say with which it will or will not work. We are glad to hear, of course, that with a small modification, it worked with the SD Sales Expandoram.

DMA bus arbitration was not included for three reasons:

1) We would have had to take out one of the other features, for example the extended address-bus feature, with the parts available at the time. There are a lot of parts on the board.

2) It wasn't clear to us from the standard whether the arbitration was a required feature to meet the standard.

3) We felt that most installations wouldn't require that more than one DMA device transfer data at a time. Note that it is possible to have our interface on the bus with another DMA interface, as long as both aren't trying to transfer data at the same time. Since the most likely candidate for the other device would be a hard disk interface, it would most likely require data transfers at too high a rate to allow our interface to transfer concurrently anyway.

We welcome any other ideas from your readers about how any of our products can be improved.

Don Tarbell
owner

DMA Operation

Up to this point, I had the controller working in programmed I/O mode, which requires active processor participation. In this mode, the operation of the double density interface is very similar to the older single density interface. The next step was to implement the DMA mode, which would allow me to run double density. Programmed I/O control requires a 4MHz system for double density, and my old Altair computer just won't run that fast.

The only change necessary to try the DMA mode was to change one equate statement in the BIOS file contained on the public domain disk. I edited the file, reassembled it, and placed a DMA version of the operating system on my disk. I hit the reset button to boot up the DMA mode operating system, hoping that the controller and my dynamic memory would co-exist peacefully. It was with much relief that I read the sign-on message from the operating system, indicating that everything had worked properly. I was running DMA mode, single density, and my memory board appeared perfectly happy.

Then I tried copying a file from one disk to another. The copy contained garbage! This was the first operation I had performed which required a DMA read operation from memory. Apparently I could do a DMA write to memory, to load the operating system from disk to memory, but could not do a DMA read to go from memory to disk.

This problem was not mentioned in the manual, so I had no clues as to what was happening. I studied the memory timing specifications to determine what was required by the memory board. Then I determined the timing signals generated by the DMA controller circuits. It would have been most helpful at this point if timing diagrams had been included with the controller documentation.

Anyway, the two signals which were causing the problem were the pSYNC signal, which defines the start of a new bus cycle, and the sMEMR signal, which identifies a data transfer from memory. The memory board requires that both the pSYNC and sMEMR signals be active to initiate a memory read cycle. The DMA controller sets pSYNC active at the start of a DMA cycle, and then turns it off as soon as the sMEMR signal goes active. My memory board would never see both of these signals active at the same time, and therefore would never initiate a read cycle. Either the memory board or the disk controller would have to modified.

I redesigned (on paper) the DMA control circuit and the memory read circuit, trying to correct this problem, but could not come up with an easy solution. Finally, I just lifted the pin which controlled pSYNC on the controller board, making the pSYNC signal true through the entire DMA cycle, and tried the copy operation again. This time everything worked fine. I could read and write disk files, and my memory board responded correctly. I am still running in this configuration. Not exactly IEEE spec, but it works.

Double Density Operation

With the DMA problem solved, I was ready to try double density operation. I formatted a disk using the disk format program supplied with the controller, and ran the disk test program. Errors and more errors! This problem was well documented and relatively easy to cure. As described in the manual, different disk drives require different pre-compensation, but I could follow the instructions provided in the manual to correct the problem. Just try different settings on the nine switches on the controller board until the errors go away. Various combinations of switch settings are listed in the manual and it is a straightforward, if somewhat time-consuming, process to configure the interface for the drives you are using. You change the switch settings, format and test a disk, and repeat the operation until the best combination is found. The switches are located at the top of the board and are easy to change.

After I had gone through this process I could operate in the double density mode without problems. The controller does seem to be a bit fussy as to the quality of the disk that I use. Some of my old single density disks will not format properly in the double density mode, but I have had no problems with certified double density disks.

Conclusion

At this point I have no major complaints with the double density interface. It is performing well, and I have over twice the disk space available compared to the older single density interface that I was using. I completely rewrote the BIOS file supplied with the controller, and achieved 596K bytes per disk, using 512 byte sectors.

My only minor complaint is in the area of IEEE compatibility. The controller is claimed to be compatible with the new IEEE specification, but only partially meets the requirements. In particular, the board operates in the DMA mode, but includes no DMA arbitration circuitry to handle multiple DMA devices on the bus. If you are going to install several DMA devices, this could lead to problems. I currently have no such plans, so I won't worry about it. Also, since no timing diagrams are included with the documentation, it is not easy to determine if the controller meets the IEEE timing specification.

The software supplied with the interface is one of the nicest features of the whole system. If I had to write all of that software from scratch, it would have slowed down the whole process considerably. Tarbell Electronics is to be congratulated for including the disk with the controller, even though I did not buy the CP/M operating system from them. ■

THINKING ABOUT WORD PROCESSING???

Alpha Data Services offers the small businessman and serious computer enthusiast a choice of the finest S-100 equipment and low cost Daisy Wheel Terminals.

We Carry a Complete Selection of Equipment By:

California Computer Systems

EXAMPLES:		LIST:	ALPHA:
2422A	FLOPPY DISK CONTROLLER WITH CP/M 2.2*	425	329
2810A	4 MHZ Z-80 CPU DMA/SERIAL/FR. PANEL	310	249
2065C	64K DYNAMIC MEMORY 200 NS	720	499
2116B	16K STATIC MEMORY 300 NS	389	269
2032B	32K STATIC MEMORY 300 NS	755	579
2200A	12 SLOT MAINFRAME 20 AMP POWER SUPPLY	435	379

COMPARE OUR LOW, LOW PRICES BEFORE PURCHASING!

DIABLO 1620

- Hytype-II Mechanism
- Mint Condition Refurbished and Tested
- 110, 150, 300, 1200 Baud
- Menu Installation Under Word-Star**

$1795.00

DTC-300S

- Reliable Hytype-I Mechanism
- Completely Refurbished and Tested
- 110, 150 or 300 Baud

$1249.00

RIBBON SPECIAL!!

- Hytype I & II Cloth and Flim Ribbons . . . black and colors **$17.50**/box of 6
- Diablo Matrix Cloth Ribbons **$25.00**/box of 6

All our terminals come with RS-232 Cable, forms tractors, print wheel, ribbon, and typewriter height roll-around stand. They are warranteed for 30 days. Crating is FREE!!

*CPM is a trademark of Digital Research
**Word-Star is a trademark of Micropro Int.

ALPHA DATA SERVICES
810 Daleview Place
Greensboro, North Carolina 27406
(919) 373-1726

Osborne's CP/M User's Guide Reviewed

by Chris Terry

Osborne CP/M User's Guide, by Thom Hogan. Osborne/McGraw-Hill, Berkeley, CA 283 pp., $12.99. 1981.

This is a reference book for the CP/M operator, as distinct from the systems programmer. It is a thorough and very readable treatment of what the system does, how to use the built-in commands and utilities supplied with the package, and contains good information on high-level languages and application software to run under CP/M. Examples of usage are plentiful, clear, and well-commented. In these respects it is first-rate.

However, every silver lining has a cloud, as they say. Here the cloud is Mr. Hogan's restricted view of what a user is or wants. Surely, a CP/M user is anyone who installs CP/M in his system? Some users may not be technically oriented and quite willing to scream for an expert if the system so much as hiccups; these are well served by Mr. Hogan. But why does he choose to limit his teaching to this particular body of users? There are many others who are not experts but are anxious to learn something about the innards of the system they use, so that they can become, if not experts, at least competent to make simple modifications—much as a car owner will learn how to change a wheel in case he gets a flat, how to check oil, water, brake and transmission fluids, or how to look for and change a dying radiator hose. This kind of user will still find Mr. Hogan a good CP/M driving instructor, but must go to other sources (such as those in the bibliography) for instruction on preventive or corrective maintenance. I think this is a pity. If Mr. Hogan had handled the technical and interfacing aspects of CP/M with the same clarity and thoroughness that he gave to the operating aspects, we might have had the long-awaited definitive book on CP/M. It would have been very big, but also very understandable. Let us all hope that he will drop the other shoe—soon!

What The Book Contains

Chapter 1 is an introduction to operating systems in general and CP/M in particular. It gives the history of CP/M, speaks of the main versions of CP/M from Digital Research and how other vendors customize these, lists the manuals supplied with versions 1.4 and 2.x, and briefly discusses the function of CP/M in a microcomputer system. There is much useful information about the differences to be expected between versions supplied by Digital Research and those supplied by other vendors (including "look-alikes" such as TP/M and CDOS). On the debit side, the section "Some Useful Terms" contained only "Bytes," "Tracks," and "Sectors." This is hardly adequate for what follows. There are twenty or so vital terms (such as "logged-in," "default," etc.) that should have been included even if it meant moving the glossary to an appendix.

Chapter 2 discusses the file-naming conventions and gives a detailed description of each of the built-in commands, including the control characters for editing the command line. On file naming, there are very clear explanations of how to use the "wild-card" characters "?" and "*", as well as explanations of why you can *not* use them in certain cases.

I particularly liked the remarks on the differences in how Versions 1.x and 2.x execute the commands, and the very clear definitions of the conventions used in this and other chapters to show precisely what the operator should type, and what the computer displays. Commands and computer responses are first shown single-spaced, as they would appear on the console, and then dissected with an explanatory comment on each part of each command or response. Additional paragraphs contain explanatory notes, or list the error messages that might appear and their probable causes. Excellent stuff. Chapters 2 through 4 and Chapter 6 all have this level of detail, and even a CP/M "expert" may learn a few things from these chapters.

Chapter 3, "CP/M Transient Commands," discusses what a program is, and shows that all commands other than the seven(six for Version1.4)built-in commands are executed by programs resident on the disk and brought into memory by typing the program name as a command. The addition of parameters to commands is discussed in some detail. The rest of the chapter discusses the housekeeping utilities STAT, PIP, ED, DUMP, and the batch processing utilities SUBMIT and XSUB. It is not clear to me why DUMP is here—it would be more logical to put it in Chapter 4 (Assembly Language utilities), but that is a very minor quibble. Again, there are very detailed, accurate, and thorough discussions of the functions of these programs, how to invoke them, and what results to expect. The discussion of PIP, in particular, is the best I have seen and includes even exotica such as the "special" PIP devices NUL:, EOF:, INP:, OUT:, and PRN:.

Chapter 4, "Assembly Language Utilities," provides full and clear descriptions of ASM, DDT, and LOAD, and exactly how to use them. Once again, it is the explana-

Chris Terry, 324 E. 35th St., New York, NY 10016.

The S-100 Bus Has Moved!

Our New Address:
14425 North 79th St.
Suite B
Scottsdale, Arizona 85260

Our New Phone Numbers:
Order Phone Number:
800-528-3138
602-991-7870

We are now in our new larger quarters
in order to serve you better with these major lines:

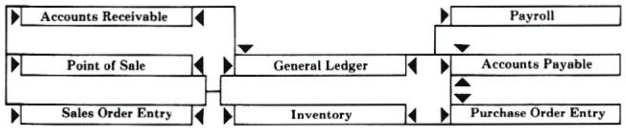
Anadex
Apple
ATV
California Computer Systems
Centronics
Cromemco
Digital Research
Electronic Control
Technology
Epson
Executive Data Systems
Godbout Electronics
(CompuPro)
Hayes Microcomputer
Heuristics
Integral Data Systems
Integrand

International Microcomputer
Corp.
Ithaca Intersystems
Maxell
Micro Pro International
Morrow Designs
(Thinker Toys)
Mountain Computer
Mullen Computer Products
North Star
Para Dynamics
Persci
Potomac Micro Magic
Qume
Sanyo
Scion
SDM Industries

SD Systems
Shugart
Soroc
SSM Microcomputer
Products
Tandon Magnetics
Tarbell Electronics
TCS
T.E.I.
Tecmar
Teletype
Televideo
Texas Instruments
Three M (Scotch)
Verbatim
Vector
Zenith

S-100 inc.
14425 North 79th St., Suite B
Scottsdale, Arizona 85260

Keep Your Balance



Accounting Plus+ brings up-to-the-minute accounting power to the microcomputer industry.
Accounting Plus is a comprehensive on-line system, with all transactions applied immediately, and results accessible without long delays associated with batch-type transactions. Additional features include:

-Supports 65K records/ data file
-8 levels password protection
-Supports 6 preprinted forms
-Supports hard disk

Our software operates under CP/M* & MP/M*, operating systems. Yesterday's dreams are available today, off the shelf.
3975 East Bayshore, Palo Alto, CA 94303, Ph. 415/969/7047

*TM of Digital Research, Pacific Grove, CA
+TM of Software Dimensions, Citrus Heights, CA.

Systems Plus Inc.

MICROSTAT™ Release 2.0

NEW RELEASE! **Just some of the new features of Microstat Rel. 2.0 include: new programs for moments about the mean, skewness, kurtosis and stepwise multiple regression, longer file names, faster sort routine, the ability to declare each data file's numeric precision and drive location plus an expanded user's manual with new appendices for the equations and file structures used in Microstat. Also included is a Data Management Subsystem for file maintenance (edit, list, destroy, augment, sort, rank-order, move and merge) plus transformations (add, subtract, multiply, divide, reciprocal, log, natural log and antilog, exponentiation and linear) that allow you to create new variables from existing variables.**

After file creation with DMS, programs for analysis include: Descriptive statistics, Hypothesis testing (mean and proportion), ANOVA (one-way, two-way, and random blocks), Scatterplots, Frequency distributions, Correlation analysis, Simple, Multiple and Stepwise Multiple Regression (including files larger than available memory), Time series, 11 Nonparametric tests, 8 Probability distributions, Crosstabs and Chi-square, Combinations, Permutations and Factorials (up to one million factorial). All program output is neatly formatted for easy use.

The price for Microstat Rel. 2.0 is $295.00 and the user's manual is available for $25.00 (credited towards purchase) and includes sample printouts with file lables that reference standard statistical texts and journals so you can compare the results from Microstat to those produced on much larger systems. Compare Microstat to any other package on the market and we think you'll agree that Microstat is the best at any price.

I(nterchange)

I(nterchange) is a general purpose file maintenance program for use with the CP/M™ operating system. Since it is a single program written in optimized Z-80™ code, it is much faster and easier to use than other file maintenance programs. Features include: DIR as usual *plus* listing all files *excluding* those with a specified character(s), ERA as usual plus *exclusive* erases. Also, a "Q" switch can be used to query each erase, a "W" allows erases of R/O files without query (normally you are queried), and an "R" switch if system files are to be included, LIST permits listings and uses TAB, WIDTH, LINES and WRAP for control, COPY as usual plus *exclusive* copies and supports the "Q", "W" and "R" switches plus an "E" switch for query on existing files, STAT with ambiguous, unambiguous and exclusive listings and produces an alphabetized listing with file length, total directory entries and space used and unused, START-END allows for copying contiguous data files, and RENAME as usual plus ambiguous renames. Other commands include: QT, DATE, TIME and SETIT (for the QT clock board) plus CLEAR, RESET, HELP and TYPE. Disk copies can even be continued after a disk full condition by simply inserting a new disk. All of this in one program without ever having to leave I(nterchange) and wait until you see the speed improvement . . .

The price for I(nterchange) is $59.95 and the manual is available for $10.00 (credited towards purchase). I(nterchange) is recommended for 32K or larger systems using CP/M™ 2.0 or later. It will not run on an 8080 CPU and only User 0 is supported.

All programs are available on 8" SD or North Star 5¼" disk. Microstat is available for North Star Basic, Microsoft's Basic-80 (Rel. 5.0 or later) or compiler Systems CBasic2. Please specify when ordering.

CP/M is a registered trademark of Digital Research.

ECOSOFT, INC.
P.O. BOX 68602
INDIANAPOLIS, IN 46268-0602
(317) 283-8883

master charge
VISA

tory text and the examples that make this chapter valuable. Where the Digital Research manual curtly informs you that a Phase error results when "label does not have the same value on two subsequent passes through the program," this chapter notes: "The label changes value during assembly. If you must reassign a value, use the SET directive. Can also be caused by a duplicate label." A duplicate is indeed the most common cause. There are similar helpful comments on all of the ASM error messages. In this section on LOAD, the text mentions that the command may specify a drive (e.g., LOAD B:FOOBAR) on which to look for the .HEX file, but does not clearly spell out that LOAD will also place the resulting .COM file on the same drive.

Chapter 5, "Other Transient Programs and CP/M," is in three sections: utility programs, high level languages and application programs. The section on utilities deals with FORMAT, COPY, MOVCPM, and SYSGEN. Examples of formatting program names are listed, a typical run sequence is shown, and the process executed by the program is described in the most general terms. But, "You probably do not care what is done, or how, so long as the disk is properly prepared." The discussion of COPY is at a similar level.

The discussion of MOVCPM and SYSGEN is, to my mind, one of the least satisfactory parts of the book. The commands and the results and messages to be expected are well described, but it is all valueless because it is out of context. You are warned that "MOVCPM moves only the raw, bare bones of the CP/M operating system. Any additional drivers or other changes you have made to the BIOS section of CP/M (see Chapter 7) will not be moved." You are *not* warned that the Digital Research version of MOVCPM contains the BIOS for the Intel MDS (Microcomputer Development System) with its disk controller, and that few hardware vendors customize this program for their own controller. Nor is any mention made of the special area in the TPA where MOVCPM places the memory image of the relocated CP/M. Discussion of these programs belongs with discussion of the BIOS, or in a separate, very detailed and painstaking chapter on the relocation of CP/M. This is a tough subject, but any user may be faced with it, if only to accomodate application programs like BYE, which require protected space above CP/M. It needs far more extensive treatment than it gets in this book.

On the other hand, the section on high-level languages is very good indeed. It gives an overview of the characteristics of a number of languages for use on a CP/M system. There is useful information on what the code looks like, what the language can and cannot do for you, and some suggestions on how to pick a suitable language for your particular field of endeavor. The languages discussed include: interpreter and compiler Basic, with considerable detail on EBasic and CBasic; the C language; Cobol; Forth; Fortran; Pascal; and PL/I-80. "How to Run" paragraphs give the principal commands and options for specific implementations of these languages.

The section on application programs contains some general notes on the running of application programs, followed by brief descriptions of several word processors (Magic Wand, Scope, Electric Pencil, WordStar).

Chapter 6, "MP/M, CP/NET, and CP/M Derivatives," describes the purpose and functions of MP/M and CP/NET. The commands and functions peculiar to these operating systems are given in detail. In the discussion of CP/M Derivatives, the emphasis is on the differences between these and CP/M. As an ignoramus in regard to these other systems, I can say only that the introductory text and the details of the commands are clear and readable. Since I know Chapters 2 through 4 to be exceptionally accurate, I would trust the accuracy of this chapter also.

Chapter 7, "Technical Aspects of CP/M," is a very mixed bag. It begins by describing the general structure of CP/M, with both tabular and diagrammatic memory maps. These are followed by brief descriptions of the CCP and the BDOS, the cold start process, and a paragraph on how to use the BDOS functions in assembly language programs. The calling sequence is shown, and Table 7-1 lists all of the BDOS functions and their codes. So far, so good.

Now, however, we come to a section on modifying the BIOS. This seems to be predicated on two assumptions: 1) that the MOVCPM supplied by the vendor has been fully customized for your system; and 2) that you can successfully overlay your current BIOS with the modified version *in the operating area*. I am far from convinced that these assumptions are justified. There is no mention of the special area of the TPA where MOVCPM places the new memory image and where the SYSGEN expects to find it, nor of how to calculate the Read offset with which DDT can bring a BIOS to the right place in that area. There are plenty of warnings and suggestions which may work if you are modifying a system of the same size, but heaven help you if you try to overlay a modified BIOS on a relocated CP/M—this chapter certainly will not. And that, with the exception of a page on the IOBYTE and the codes for device names recognized by STAT, is it. There's nothing about directory entries, nothing about how CP/M allocates disk space, nothing technical at all.

Contrasting this skimpy and (I think) misleading material with the detailed content and high level of teaching competence found in Chapters 2 through 6, I found this chapter bitterly disappointing. But perhaps I am being unfair in asking my driving instructor to teach me maintenance as well.

Chapter 8, entitled "Putting It All Together: The Systems Approach": was missing from my proof copy, but will contain System Recommendations, Procedure Recommendations, and The Systems Approach.

Six appendices contain a CP/M Command Summary, ASCII character codes, Comparison of CP/M Versions 1.3, 1.4 and 2.0 in tabular form, CP/M Prompts, Diskette Selection (the size, type, and sectoring for various standard systems and controllers), and an extremely useful annotated bibliography. And, to restore one's faith in mankind, an excellent index. I chose about thirty items to spot-check, found index entries for them without difficulty, and the page references were accurate. There are positively *no* circular cross-references (Basilisk—see Weasel, Weasel—see Basilisk) which still occasionally creep into the works of publishers less careful than Osborne/McGraw-Hill. The book was written and edited with the aid of WordStar, and has been set in a very pleasing and legible typeface. ■

C Compiler only $75

We have re-written Small-C as published by Ron Cain in the May, 1980 issue of Dr. Dobbs. The Code Works C compiler (CW/C) includes these additional features:

- **Structures and unions**
- **For, switch/case, do-while**
- **Multidimensional arrays**
- **Conditional compilation (#ifdef,etc.)**
- **Assignment operators, e.g. x += 10;**
- **Can declare complex types, e.g. int (*fp)[5];**
- **User supplied I/O buffers of any size**
- **Dynamic storage allocation (alloc and free)**
- **Command line arguments using argv and argc**
- **Improved error handling**

CW/C is a proper subset of the full C language. We do not have: float, double, long, unsigned or short data types; static; initializers; sizeof; typedef; "?:"; casts; bit fields; goto; #undef, #if, #line.

CW/C generates assembly language source code that is then assembled using ASM or MAC. CW/C supports inline assembly language with the #asm ... #endasm preprocessor commands. Requires 56K 8080 or Z80 CP/M system. Distributed on single-density 8" disk or Northstar double density CP/M 5" disk. Includes an excellent User Manual, the executable CW/C compiler, runtime library, and several useful example programs written in C.

THE CODE WORKS

CW/C is $75, including shipping in the US and Canada.
CA residents add 6% tax. Visa and MasterCard welcome.
CP/M is a registered trademark of Digital Research.

Box 550, Goleta, CA 93116 805-683-1585

PROBLEM:

How to mate your CP/M® or ISIS® system to the

Z-8000

(without losing all your present capabilities)

SOLUTION:

ZEX is the software connection which permits Z-8000 applications to run under your current operating system, resulting in portable 16-bit software which runs under CP/M or ISIS! And ZEX is user-configured for any Z-8000 alternate bus master, so that it can support your prototype hardware, as well as commercially available Z-8000 boards.

ZAS is a complete Z-8000 Development Package, which includes a powerful relocatable cross assembler (ZAS), a flexible object task builder (ZLK), and an absolute object loader (ZLD), as well as the ZEX run time module. The package was developed specifically for the Z-8000, and supports the complete Zilog instruction syntax for both the Z8001 and Z8002, including support for mixed segmented and non-segmented code.

WESTERN WARES

ZAS Package	CP/M®	**$395**
	ISIS-II®	**$495**
Manual Only		**$25**

Supplied on Single Density 8" Disk

CP/M· Digital Research, Inc. ISIS-II· Intel Corp.

Box C Norwood, CO 81423 (303)327-4898

Now

tiny c Soars!

...with tiny-c two — the compiler

Tiny-c two is ten times faster than tiny-c one, with many features, including long (32 bit) integers, lots of new operators, and re-directable and direct access input/output. Viable for professional work, either systems programming or business applications.

It comes with a UNIX™ style command interpreter called the "tiny-shell"™. Every compiled tiny-c program becomes a new shell command. Commands can have arguments, and dash (-) options, just as real UNIX shell commands do. The < and > input/output redirection operators are supported.

Fifty standard library functions, and readily extended. The input/output functions are UNIX styles, including fopen, fprintf, etc. Both ascii and raw (binary) input/output are supported.

Package is portable. Bringing it up on a new processor or new operating system should take just days. And as usual with tiny-c products, all the source code is included.

Tiny-c two is available now on standard 8" CP/M.

$250.00 - Includes Owners Manual and Disk
Manual Only $50.00
(20% Discount to tiny-c one owners)

The original tiny-c ONE is still available on a wide variety of cassettes and diskettes. This version is an interpreter, complete with a Program Preparation System. Disk or cassette versions $100 (this price includes the Owners Manual, available separate-at $50). Disks: CP/M, Apple DOS 3.2, H8/89 HOS, PDP-11, Flex 2.0, Northstar, CDOS. Cassettes: KIM, SYM, TRS-80, Tarbell, Cuts.

tiny c

Call or write tiny-c associates, P.O. Box 269, Holmdel, N.J. 07733 (201) 671-2296. You'll discover tiny-c is flying higher and faster.

New Jersey residents include 5% sales tax. Visa or Master Card accepted. Include charge plate number with order.

UNIX is a trademark of Bell Laboratories, Inc.
tiny-c and tiny-shell are trademarks of tiny c assoc.

A Disk Alignment Routine

by Lorin S. Mohler

This routine (written in the 'C' language,) is handy to use during disk drive alignment. It is for systems using the single-density Tarbell interface.

If you do your own disk drive alignment, you will find this program very handy.

Install the diskette containing ALIGN.COM in a working drive on the system. Execute ALIGN. Turn the disk drive power off and install the drive to be aligned. The drive must be configured as A, B, C or D. Turn the drive power on and insert an alignment diskette. Select the drive with the 'D' command. Select the track with the 'T' command. Load and unload the head with the 'L' and 'U' commands. Follow the recommended manufacturers alignment procedure for your drive. The head load or unload will be maintained at 'command:' only.

```
/*ALIGN. C

#define READ 0x8C
#define SEEK 0x12
#define BUSY 0x01
#define LOAD 0x0001
#define UNLOAD 0x0000
#define SECT 0xFA
#define DDATA 0xFB
#define DEXT 0xFC
#define WAIT 0xFC
#define DCOM 0xF8
#define DSTAT 0xF8

int head_load;                      /* global */

main()
{
char c;

head_load = UNLOAD;                 /* initialize head condition */

  printf("rev. 9/23/80\n\n");
  while(1) {
      printf("command: ");          /* issue a prompt */

          while ( !kbhit())         /* maintain head */
                  head();

          switch(c = toupper(getchar())) {        /* get command */

          case 'L':       printf("oad\n");
                          head_load = LOAD;
                          break;
          case 'U':       printf("nload\n");
                          head_load = UNLOAD;
                          break;
          case 'T':       printf("rack #: ");
                          seek();
                          break;
          case 'D':       printf("rive: ");
                          select();
                          break;
          case 'B':       printf("ye...\n");
                          exit();
          default:        printf(" ?\007");
                          pcmnds();


                  }
          }
}
```

```
seek()                      /* move head to selected track */
{
  char c[80];
  int track;

  track = atoi(gets(c));                          /* get track number */
  if (track < 0 || track > 76) {                  /* check legallity */
          printf(" ?\007: range 0-76\n");
          return(-1);                             /* error flag */
  }
  outp(DDATA,track);                              /* set track */
  while (inp(DSTAT) & BUSY)
          ;
  outp(DCOM,SEEK);
  inp(WAIT);
  printf("\n");
  return(0);
}

select()                            /* select the drive */
{
  char drive;

  drive = toupper(getchar());                     /* get drive */
  if (drive < 'A' || drive > 'D') {               /* check legallity */
          printf(" ? \007: range A-D\n");
          return(-1);                             /* error flag */
  }
  outp(DEXT, ~(drive-'A') << 4 | 2);              /* set drive */
  printf("\n");
}

head()                          /* maintain the head load or unload */
{
  if (head_load) {
          outp(SECT,1);             /* sector 1 */
          outp(DCOM,READ);          /* read loads the head */
  }
}

pcmnds()                            /* print the legal commands */
{
  printf (": (l)oad & (u)nload head, (t)rack, (d)rive & (b)ye\n");
}
```

This program was compiled using the BD Software C compiler written by Leor Zolman. The User's Guide that comes with the diskette is excellent. Also be sure to get a copy of *The C Programming Language* by Brian W. Kernighan and Dennis Ritchie (Prentice Hall 1978) for a proper description of the language. ■

Lorin S. Mohler, PO Box 8340, Anahiem, CA 92802

Desk Main/Frame

Desk Main/Frame

LOW COST & ATTRACTIVE STYLING

- MAIN/FRAME INTEGRATED INTO FURNITURE QUALITY DESK
- ELECTRONICS PACKAGE SLIDE MOUNTED FOR EASY ACCESS
- SUPPORTS TWO 8" FLOPPY DRIVES FROM SEVERAL MANUFACTURERS (DRIVES NOT INCLUDED)
- 10 SLOT MOTHERBOARD INCLUDES CONNECTORS
- POWER SUPPLY FOR DRIVES AND CARDS
- DESK AND MAIN/FRAME AVAILABLE SEPARATELY
- MATCHING PRINTER DESK AVAILABLE

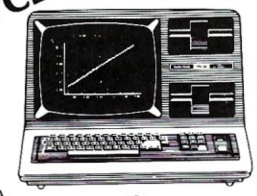
WRITE OR CALL FOR OUR BROCHURE WHICH INCLUDES OUR APPLICATION NOTE: 'BUILDING CHEAP COMPUTERS'

INTEGRAND

8474 Ave. 296 • Visalia, CA 93277 • (209) 733-9288
We accept BankAmericard/Visa and MasterCharge

Calling Carrier Dealers!

Why limit yourself! Our CL4M Program is many times MORE POWERFUL than the Carrier program!



- 256 Zones, 256 Systems and Block Load AUTOMATICALLY
- ASHRAE 1977 Methods • 5 Load Walls in Each Zone
- Shading Projections • Comparable to Main Frame Programs
- For TRS-80 Models I, II & III • **Now for CP/M Machines Too**

Commercial Clg/Htg Loads for Microcomputers

MC↑2™ PROGRAMS

McClintock Corp.
P.O. Box 430980
Miami, FL 33143
(305) 666-1300

PLEASE TELL ME MORE ABOUT YOUR POWERFUL CL4M PROGRAM.

Name ______ Company ______ Address ______ Phone ______ City ______ State ______ Zip ______ MS 11

Bring the flavor of Unix To your Z80-based CP/M system with Unica

"Unicum: a thing unique in its kind, especially an example of writing. Unica: the plural of unicum."

The Unica: a unique collection of programs supporting many features of the Unix operating system never before available under CP/M. The Unica are more than software tools; they are finely crafted instruments of surgical quality. Some of the Unica are:

bc	-	binary file compare
cat	-	catenate files
cp	-	copy one or more files
dm	-	disk map and statistics
hc	-	horizontal file catenation
ln	-	create file links (aliases)
ls	-	directory lister
mv	-	move (rename) files, even across users
rm	-	remove files
sc	-	source file compare, with resynchronization
srt	-	in-memory file sorter
sr	-	search multiple files for a pattern
sp	-	spelling error detector, with 20,000 word dictionary

Each Unicum understands several flags ("options" or"switches") which control program alternatives. No special"shell" is needed; Unica commands are typed to the standard CP/M command interpreter. The Unica package supports several Unix-like facilities, like filename user numbers:

```
     sc data.bas;2 data.bas;3
```

(compares files belonging to user 2 and user 3); Wildcard patterns:

```
     rm *tmp* -v
```

(types each filename containing the letters TMP and asks whether to delete the file);
I/O redirection:

```
     ls -a ►list
```

(writes a directory listing of all files to file"list");
Pipes:

```
     cat chap* ! sp ! srt ►lst:
```

(concatenates each file whose name starts with"chap", makes a list of mispelled words, sorts the list, and prints it on the listing device).

The Unica are written in XM-80, a low level language which combines rigorously checked procedure definition and invocation with the versatility of Z80 assembly language. XM-80 includes a language translator which turns XM-80 programs into source code for MACRO-80, the industry standard assembler from Microsoft. It also includes a MACRO-80 object library with over forty "software components", subroutine packages which are called to perform services such as piping, wildcard matching, output formatting, and device-independent I/O with buffers of any size from 1 to 64k bytes.

The source code for each Unicum main program (but not for the software component library) is provided. With the Unica and XM-80, you can customize each utility to your installation, and write your own applications quickly and efficiently. Programs which you write using XM-80 components are not subject to any licensing fee.

Extensive documentation includes tutorials, reference manuals, individual spec sheets for each component, and thorough descriptions of each Unicum.

Update policy: each Unica owner is informed when new Unica or components become available. At any time, and as often as you like, you can return the distribution disk with a $10 handling fee and get the current versions of the Unica and XM-80, with documentation for all new or changed software.

The Unica and XM-80 (which requires MACRO-80) are priced at $195, or $25 for the documentation. The Unica alone are supplied as *.COM executable files and are priced at $95 for the set, or $15 for the documentation. Software is distributed on 8" floppy disks for Z80 CP/M version 2 systems.

Knowlogy
"Shaping Knowledge for Evolving Worlds"
P.O. Box 283
Wilsonville, Oregon 97070

Visa/Mastercard customers call (503) 635-5701 after hours for next day shipment.

CP/M is a trademark of Digital Research; Unica is a trademark of Knowlogy; Unix is a trademark of Bell Telephone Labs; XM-80 is a trademark of Scientific Enterprises; Z80 is a trademark of Zilog Inc.

SOFTWARE DIRECTORY

Program Name: COMPRESS
Hardware System: CP/M
Minimum Memory Size: 16K
Language: 8080 Assembly
Description: COMPRESS is a group of four programs which perform data compression on ASCII data. There are two .COM files which will compress and decompress disk files, and two .REL files which are in subroutine form and act on memory use. Compression varies depending on the contents of the source file, but normally is in the range of 30% - 80%. Works on all 7 bit ASCII data.
Release: July 1981
Price: $50
Included with price: 8" CP/M disk with REL, COM, and DOC files.
Where to purchase it:
New Jersey Software Services
6 Village Circle
Westfield, NJ 07090

Program Name: SCREENMASTER
Hardware System: CP/M compatible. Any dumb terminal.
Minimum Memory Size: 48K recommended.
Language: CBasic-2.
Description: A complete screen handler facility, provided in source code for inclusion in programs requiring full/multi screen input. GOSUB SCREENMASTER. Upon return, multiple screens of validated input are available, in memory, for further use by the programmer. Programmer may insert CBasic-2 code at any of the many exits to affect any degree of editing or control, overriding SCREENMASTER editing if desired. The programmer and, optionally, the terminal user have commands: GOTO (field) n, BACK/FORWARD n (fields), NEXT/PRIOR (screen), PRINT (screen), as well as SUBMIT and ABORT. Utilities are provided to create and test screens as well as to configure any dumb terminal.
Release: August 1981
Price: $195 Manual alone $25 Demo disk $10 additional.
Included with price: 90 page user manual, floppy disk with source code, demos and utilities.
Where to purchase it:
Marketing Essentials, Inc.
206 Mosher Ave.
Woodmere, NY 11598
(212) 580-3589

Program Name: FORTH by Timin Engineering, Release 2
Hardware System: 8080/8085/Z-80 CP/M or CDOS
Minimum Memory Size: 24K
Description: Enhanced version of FIG FORTH. A FORTH style editor with twenty commands is included, as well as virtual memory sub-system for disk I/O. The user may also make permanent additions to the resident FORTH vocabulary. A Z-80/8080 assembler is also included, allowing the user to create new FORTH definitions which compile directly into machine code. The IF...ELSE.., BEGIN...UNTIL, and BEGIN... WHILE... control structures may be included in assembler definitions. Other enhancements include an interleaved disk for disk access. A 1024 byte disk block may be read or written in as little as 1/6 second. Eight of these blocks are maintained in RAM for immediate access and automatically swapped with others on the disk as they are needed.
Released: December 1980
Price: $95, IBM compatible 8" SD disk (other formats $110). Manual only: $20, applicable toward purchase on disk. California residents add 6% sales tax.
Included with price: CP/M compatible disk, user's manual and shipping by mail in U.S.
Authors: Dr. Mitchell E. Timin, and FIG
Where to pruchase:
Timin Engineering Co.
9575 Genesee Avenue, Suite E-2
San Diego, CA 92121
(714) 455-9008

Program Name: CONST
Hardware System: North Star & Apple
Language: Basic
Description: This program was written to do quantity and sizing take-offs for residential and small commercial structures. To operate the program, the user has only to answer questions concerning room sizes and type of construction.
Release: July 1981
Price: $75; listing only $60.
Included with price: Diskette, On-line Documentation, Support
Where to Purchase it:
Computing Interface
1918 Carnegie Lane #C
Redondo Beach, CA 90278

INVENTORY $400*

for OSBORNE/McGRAW-HILL'S CBASIC ACCOUNTING PACKAGE

- SUPPLIED IN SOURCE (.BAS) FORMAT
- COMPREHENSIVE REPORTS INCLUDE
  REORDER, PRICE LISTS, DETAILED INVENTORY
- DIRECT ENTRY, WITH DETAILED LISTINGS, OF
  CUSTOMER SALES & RETURNS AND VENDOR RECEIPTS & RETURNS
- ALLOWS INVENTORY TO BE RESERVED AND DISTRIBUTED TO MULTIPLE LOCATIONS
- ALSO SUPPORTS CHAIN STORE SALES WITH DISTRICT LEVEL BILLING
- PLUS MUCH, MUCH MORE!

HARDWARE REQUIREMENTS

8080 or Z80 CPU	
48K ram	132 COLUMN LINE PRINTER
24x80 CRT TERMINAL	DUAL 8 INCH DISK DRIVES

SOFTWARE REQUIREMENTS — *(ALL AVAILABLE AT ADDITIONAL COST)*
CBASIC2, VSORT, OSBORNE G/L, A/R, A/P

→ Complete Hardware/Software Systems also available ←

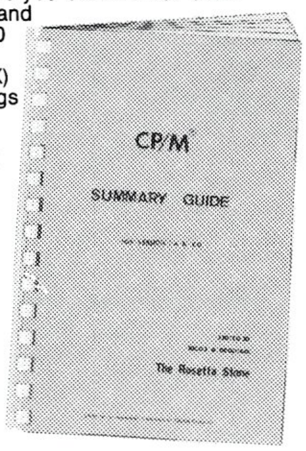
Executive Data Systems
541 R Kenilworth Blvd.
Kenilworth, New Jersey 07033
(201) 241-4887

*Dealer Inquires Invited

COD

CP/M SUMMARY GUIDE

Tired of fanning through your CP/M manuals or writing notes that remind you of the commands, functions and error codes? Well it's about time you ordered our CP/M Summary Guide! Spiral bound and handy to hold, our guide is a 60 page booklet summarizing the features of CP/M (Ver. 1.4 & 2.X) and 2 totally alphabetical listings of the commands, functions, statements and error codes of MICROSOFT BASIC-80 Ver. 5.0 and CBASIC™ -2. Areas summarized are in table form and include all direct and transient commands plus MAC™, DESPOOL™ and TEX™. Our booklet is a much needed supplement to any of the literature currently available on CP/M and has been recommended by Digital Research.



P.S. Over 4000 users can't be wrong!

Ask your local computer store for our guide or send $6.95 plus $1.00 (postage and handling) to:

THE ROSETTA STONE, P.O. BOX 35, GLASTONBURY, CT 06025 (203/633-8490)

Name ____________________

Street ____________________

City __________ **State** ______ **Zip** ______

CP/M™, DESPOOL™, MAC™ are registered trademarks of Digital Research. CBASIC™ is a registered trademark of Compiler Systems.

INTRODUCTORY OFFER!

*Turn your Micro-Computer into a Mini-Computer. Try the world's #1 programming language-**COBOL!** Finally at a price you can afford and with no risk!*

Introducing...

NPS-MICRO-COBOL

This is the Naval Post Graduate School Cobol that you've heard so much about. Designed to pass the stringent government Hypo-COBOL tests used by GSA in their Compiler Certification Program. This is the first public release of version 2.1. This is an elaborate ANSI-COBOL subset. Comes complete with users manual in DeLuxe three-ring binder.

- **Perfect for learning COBOL.**
- **Perfect for teaching COBOL**
- **FREE sample programs included**
- **Runs in 24K**
- **Requires 8080, Z-80®, or 8085 and standard CP/M® system**
- **Provided on standard 8" disk or Northstar Double Density CP/M 5"**

Only $69.95!

FREE ALGOL INCLUDED!

FREE BONUS. All purchasers receive a FREE copy of NPS-ALGOL at no extra cost. A favorite language in Europe, ALGOL is the original structured language. Comes with FREE sample programs.

MONEY-BACK GUARANTEE. *If you're not completely satisfied with this software. You may return it within fifteen days for any reason and get a full refund.*

Send Check, Money Order or Credit Card information and order a copy today! Please add $2.50 shipping and handling on all orders.

Credit Card buyers: For Extra Fast service call (415) 527-8717

**Order from: The Software Review
704 Solano Avenue, Albany, CA 94706**

Yes, I want to run COBOL on my system! Enclosed find $69.95 plus $2.50 shipping/handling (California residents please add appropriate sales tax). I will receive the NPS-COBOL system plus a FREE copy of NPS-ALGOL. I understand that I may return the software within 15 days if not completely satisfied for a full refund.

NAME ____________________

COMPANY ____________________

STREET ____________________

CITY ____________________

STATE __________ ZIP __________

AMOUNT ENCLOSED $ __________

Disk size desired: 5" 8"

☐ Check Enclosed ☐ VISA

☐ UPS C.O.D. ☐ Mastercharge

Card number ____________________

Expiration Date ____________________

Signature ____________________

☐ Check here for more information

CP/M is a trademark of Digital Research and Z-80 is a trademark of Zilog.

CP/M® SOFTWARE

For The Small Computer System

NEW PRICES
NEW
SOFTWARE

*** * * *DISCOUNTS* * * ***

- IMS CHECKED (√) ITEMS—BUY TWO CHOOSE A THIRD ONE FREE
- CHOOSE 1 BOOK LISTED FOR EACH ITEM ORDERED OVER $200

*** * * * *LIMITED OFFER * * * * ***

ACCOUNTING

		COMPLETE	MANUAL ONLY	SYSTEM REQUIREMENTS
GENERAL LEDGER	√IMS	$540	$40	1,7
ACCOUNTS RECEIVABLE (B/F)	√IMS	540	40	1,7
ACCOUNTS RECEIVABLE (O/I)	√IMS	540	40	1,7
ACCOUNTS PAYABLE	√IMS	540	40	1,7
PAYROLL	√IMS	540	40	1,7
JOB ACCOUNTING	√IMS	540	40	1,7
FUND ACCOUNTING	√IMS	540	40	1,7
CASH RECEIPT/DISBURSEMENT	IMS	200	40	1,7
DEPRECIATION SCHEDULE	IMS	100	40	1,7

GENERAL BUSINESS SYSTEMS

MFG. INVENTORY CONTROL	IMS	$600	$75	1,7
WHOLESALE/RETAIL DISTRIBUTION	IMS	600	75	1,7
MEDICAL/DENTAL SYSTEM	IMS	600	75	1,7
STUDENT RECORDS/SCHEDULING	IMS	540	50	1,7
PROPERTY MANAGEMENT	ASC	1095	75	1,7,12
PROPERTY MANAGEMENT	A-T	640	50	1,7
PROPERTY ANALYSIS SYSTEM	A-T	245	50	1,7
INSURANCE AGENCY SYSTEM	Univar	940	75	1,7,9
MONEY MAESTRO-HOME BANKING	InnoSys	190	25	1

MODELING, PLANNING & ANALYSIS

MINI MODEL™	Financial Plan Assoc.	$490	$60	1,7
STATPAK	Northwest Analytical	490	50	1,8
SHORTAX	Syntax	490	40	2,5,8
MILESTONE™	Organic Software	290	40	1
MICROSTAT	Ecosoft	290	25	15
SUPERCALC™	Sorcim	285	—	1,6
muSIMP/muMATH	Microsoft	245	30	2,5,9
MAGIC WORKSHEET	Structured Systems	235	35	1,2
WORKSHEET	SoHo Group	200	30	1,9,7 or 8
TARGET™ PLANNER	AMS	195	25	4,5
FORECASTER	Software Establishment	145	25	2,5,7,9
REPORT WRITER™	Carolina Bus. Sys.	145	45	1,8

TIME, BILLING & SCHEDULING

LEGAL TIME ACCOUNTING	Univar	$945	$60	1,7,9
LEGAL BILLING & TIMEKEEPING	Microcraft	650	60	1,7,9
PROFESSIONAL TIME ACCOUNTING	Asyst	590	40	1,7,9
CLIENT BILLING	Serendipity	590	40	1,4,7,9
DATEBOOK II™	Organic Software	290	40	1,9
TIME MASTER	Southern Digital	195	25	1,7
CONSULTANT'S BILLING	Cedar Hill	100	25	1,7

LANGUAGES

µAPL	Softronics	$345	$50	2,5
BASIC COMPILER	Microsoft	350	—	2,5,9
BASIC - 80	Microsoft	325	—	2,5,9
XYBASIC	M. Williams Co.	450	25	2,5,9
XYBASIC COMPILER	M. Williams Co.	450	25	2,5,9
S - BASIC™	Topaz	290	35	2
CBASIC - 2™	Compiler Systems	120	20	5
TINY C - II	Tiny C Assoc.	245	50	2,5,13
TINY C	Tiny C Assoc.	100	50	2,5,13
COBOL - 80	Microsoft	690	—	2,5,9
NEVADA COBOL	Ellis	140	25	2,5
FORTH (Z80)	Lab. Microsystems	45	25	2,6
FORTRAN - 80	Microsoft	430	—	2,5,9
muLISP	Microsoft	200	—	2,5,9
PASCAL Z	Ithaca	390	30	3,17
PASCAL/MT + ™	MT Microsystems	485	30	3,6
PASCAL/M™	Sorcim	220	30	3,9,12
PL/1 - 80™	Digital Research	490	40	2,6,9,11

MAILING LIST

POSTMASTER	Teratek	$145	$25	2,5,7
MAILING LIST MANAGEMENT	IMS	130	40	5,7
MAIL - MERGE™ FOR WORDSTAR	Micropro	130	30	5,7,9
NAD	Structured Systems Group	110	25	2,5,9
SUPER - M - LIST	Supersoft	100	25	2

TELECOMMUNICATIONS

IE/MODEM™	Information Eng	$490	$35	9
ASCOM	DMA	170	25	9
TERM	Supersoft	145	25	2,13
COMMX	Hawkeye	70	25	9,18

* * * * * * * * * * *

ORDER INFORMATION

- **CALL TOLL FREE 800 538-3160**
- CALIF (408) 996-8560
- OVERSEAS AND C.O.D.—ADD $10 + SHIPPING/HANDLING
- SHIPPING/HANDLING—ADD $2.50 PER ITEM
- CALIF. RESIDENTS—ADD 6-1/2% SALES TAX
- PAYMENT—PREPAY, CHECK, MONEY ORDER, UPS C.O.D.
- ALL ITEMS SUBJECT TO AVAILABILITY
- ALL PRICES SUBJECT TO CHANGE WITHOUT NOTICE
- ALL ORDERS SHIPPED UPS, F.O.B. SAN JOSE
- MOST DISK FORMATS AVAILABLE
- MANUAL ORDERS WILL BE APPLIED TO SOFTWARE PURCHASES WITHIN 90 DAYS
- THE SALE OF EACH SOFTWARE CONVEYS A LICENSE FOR USE ON ONE SYSTEM ONLY

WORD/TEXT PROCESSING

SELECT™ w/ SUPERSPELL™	Select Info	$590	$50	2,5,9
VTS/80™ w/ SPELLRITE	Kmega	585	75	2,5
WORDSTAR™	Micropro	440	60	5,9,10
SPELLBINDER™	Lexisoft	490	50	2,5,9,14
DATASTAR™	Micropro	340	60	5,9,10
SPELLGUARD™	ISA	285	25	2,5,14
SPELLSTAR	Micropro	240	60	5,9,14
LETTERIGHT	Structured Systems Group	190	25	1,7,9
MINCE	Unicorn	120	30	2,5,10
SCRIBBLE	Unicorn	120	30	3,5,10
TEXTWRITER III™	Organic Software	120	40	2,5,14
TEX	Digital Research	100	20	5,9,11

DATA/FILE MANAGEMENT

FMS-80	System Plus	$975	$60	1,9
CONDOR - II	Condor Computer	940	50	1
ACCESS/80	Friends Software	780	50	3,5
dBASE II	Ashton-Tate	680	50	1,9
PRISM/IMS™	Micro Appl Group	485	55	1,7,9
SELECTOR IV™	Micro-Ap	540	35	1,3,7,10
RADAR - DATA ENTRY	Southern Computer	485	30	9,10
T.I.M.	Innovative Software	390	30	1,15
CONFIG. BUSINESS SYSTEM	DMA	390	50	1,9
MAGSAM IV™	Micro Appl Group	290	25	1,7,9
MICRO B +	Fair Com	255	30	3,9,16
ANALYST	Structured Systems Group	240	25	1,7,9
FABS	Computer Control	190	25	
WHATSIT?™	Computer Headware	170	30	2,5,7

SOFTWARE DEV TOOLS/UTILITIES

ZSID	Digital Research	$100	$25	5,9,11
MAC	Digital Research	90	25	5,9,11
SID	Digital Research	75	25	5,9,11
DESPOOL	Digital Research	50	—	5,9,11
MACRO - 80	Microsoft	195	—	2,5,9
EDIT - 80	Microsoft	110	—	2,5,9
RAID - DEBUGGER	Southern Computer	240	25	2,9
IBM2CPM	Precision Comp. System	100	—	2,5
VEDIT	CompuView Products	125	25	2,5
DISK DOCTOR	Supersoft	95	20	2
DIAGNOSTIC II	Supersoft	95	25	2
FILE INDEX	Fisher	25	—	

SORTING

SUPERSORT I™	Micropro	$220	$60	5,9
ULTRASORT II™	Computer Control	150	25	7,9
QSORT	Structured Systems Group	95	25	2,5,9

CP/M

CP/M 2—TRS 80 MODEL II™	P&T	$185	$40	

BOOKS

CP/M HANDBOOK	Sybex	$14.95
CP/M USERS GUIDE	Osborne	12.99
CP/M PRIMER	Sams	11.95
HOW TO GET STARTED WITH CP/M	Dilithium	9.95
USING CP/M - SELF TEACHING	Wiley	9.95
CP/M SUMMARY GUIDE	Rainbow	6.95
MICROSOFT BASIC	Dilithium	12.95
FIFTY BASIC EXERCISES	Sybex	12.95
BASIC PRIMER	Sams	10.95
BASIC BUSINESS SOFTWARE	Sams	9.95
MICROSOFT FORTRAN	Dilithium	15.95
PASCAL PRIMER	Sams	16.95
PASCAL HANDBOOK	Sybex	14.95
PASCAL USERS MANUAL AND REPORT	Jenson/Wirth	14.95
INTRO TO PASCAL	Sybex	14.95
PROGRAMMING THE Z80	Sybex	14.95
YOUR FIRST COMPUTER	Sybex	7.95

SYSTEM REQUIREMENTS—All software requires a CP/M operating system as well as indicated requirements listed below:

1. RECOMMENDED SYSTEM—CP/M 1.4 OR HIGHER, 48K MEMORY, 200K DUAL DISKS, 24 × 80 CURSOR ADDRESSABLE TERMINAL, 132 COLUMN PRINTER
2. 48K MEMORY OR HIGHER
3. 56K MEMORY OR HIGHER
4. 64K MEMORY OR HIGHER
5. CP/M 1.4 OR HIGHER
6. CP/M 2.0 OR HIGHER
7. CBASIC-2
8. BASIC-80 (MBASIC) 4.51
9. SIGNED LICENSE REQUIRED BEFORE SHIPMENT
10. CURSOR ADDRESSABLE TERMINAL
11. CP/M SERIAL # REQUIRED
12. SPECIFY Z80, 8080, OR CDOS
13. SUPPLIED IN SOURCE CODE
14. COMPATIBLE WORD/TEXT PROCESSOR
15. BASIC-80 (MBASIC) 5.0 OR HIGHER
16. MUST SPECIFY HOST APPLICATION LANGUAGE
17. REQUIRES Z80 CPU
18. PRICE INCLUDES TWO CONFIGURATIONS

CP/M IS A REGISTERED TRADEMARK OF DIGITAL RESEARCH
Z80 IS A REGISTERED TRADEMARK OF ZILOG INC.
TRS 80 MODEL II IS A REGISTERED TRADEMARK OF TANDY CORP.
™ DESIGNATES NAME IS TRADEMARKED BY MFG.

CORNERSTONE SOFTWARE

PO BOX 5151 • SAN JOSE, CA 95150

TOLL FREE 800 538-3160 CALIF (408) 996-8560

DEALER INQUIRES INVITED

Software Directory, continued...

Program Name: Programmer's Apprentice
Hardware System: 8080/Z80, CP/M
Minimum Memory Size: 56K of RAM
Language: MBasic
Description: A program development that uses a macro-like language to define standard routines used by its code generator to create MBasic source code programs. Creates fully debugged programs in MBasic to provide screen prompted data input, data base management, file maintenance, and report generation. It is a visually oriented system designed to increase productivity and ease development of interactive application and report programs by performing most of the routine drudgery of programming.

To generate a program, first one creates the screen or report template on the CRT via the built-in screen editor. Then the definition modules use the screen/report template to define the various fields' attributes, eliminating most input errors, and selecting which fields will be the record control keys. Finally, the actual MBasic source code is generated.
Release: October 1981
Included with price: Manual, diskette, software including 80 column report generator.
Where to purchase it:
The Software Group
10471 S. Brookhurst
Anaheim, CA 92804
(714)535-5274

Program Name: WORD-C1, a text formatter
Hardware System: CP/M 2.2, an 8" dual diskette.
Minimum Memory Size: 60K
Language: Compiled
Description: A text formatter to prepare letters, memos, reports, manuals, documents and books. Commands set page length, line width, skip pages and text, indent, center text, etc. Line spacing, filing, adjusting, margin right justification and page numbering are automatically controlled. WORD has no limit to text size, no special hardware or modification. The Mail merge option lets you merge text with data files created by IDM-C1.
Released: September 1981
Price: $85
Included with price: 8" diskette, user's manual & postage.
Where to purchase it:
Micro Architect Inc.
96 Dothan Street
Arlington, MA 02174
(617)643-4713

Program Name: Disc-tionary
Hardware System: CP/M
Minimum Memory Size: 32K
Language: Z80 machine code
Description: High speed text proofreader. A thirty page document can be checked for errors in less than two minutes. Each unrecognized word can be added to the Disc-tionary, rejected, ignored etc. with a

DON'T MISS PHILADELPHIA AREA COMPUTER SHOW

Sponsored by Philadelphia Area Computer Society

PACS 81

NOVEMBER 12-14
Philadelphia Sheraton Hotel
17th & J. F. Kennedy Boulevard

Featuring Applications of Computers in

- **EDUCATION**
- **BUSINESS**
- **LEISURE ACTIVITIES**

PACS-81 covers a broad spectrum of interest among computer USERS. A special area of the Exhibit Hall will be reserved for

APPLICATIONS IN EDUCATION

and Computer Assisted Instruction. The most comprehensive educational exhibit to date.

BUSINESS AND HOBBY

application will also be exhibited. Commercial and Proprietary Software and Hardware will be on display including Special Exhibits on

COMPUTER MUSIC AND ROBOTICS.

CONFERENCE:
November 12-14
(Thurs., Fri., Sat.)
Speakers, Workshops and Seminars.

EXHIBITS:
November 13 (Fri.) 11-9
November 14 (Sat.) 10-6

ADMISSION: $5

EXHIBITOR BOOTHS AVAILABLE

CONTACT:
HOWARD LUBERT
5127 Windward Lane
Bensalem, Pa. 19020
(215) 245-0554

Software Shops

—Arizona—

Creative Software Systems: Systems integration and custom software (BASIC, PASCAL, Z-80 assembler). Small business and word processing systems. 632 Camelot Dr., Sierra Vista, AZ 85635.
Phone (602)458-6063.

—California—

Clear Systems
309 Santa Monica Blvd., # 404
Santa Monica, CA 90401.
(213)394-7740.
(making complexity serve simplicity)

—Colorado—

Random Factors LTD.: Industrial test, control & data acquisition — Hi speed & accuracy for S100 & STD-BUS. From software to complete systems. W.K. Borsum, P.E., Random Factors LTD. Castle Rock, CO 80104. (303) 688-5338.

Nelson Engineering: We write applications software for all micro-based systems in Asembly language, Basic, and Pascal. (213) 390-2963; 13450 Maxella Ave. G185 Suite 142, Marina Del Rey, CA 90291.

—Massachusetts—

MICROFT INC.: Customization of CP/M-80, MP/M, CP/M-86 and other operating systems. Full range of consulting services in microsystems software (systems, utilities applications), product selection, hardware. Contact: Tom Campbell, Chief of Technical Staff, P.O. Box 128, E. Falmouth, MA 02536. Phone (617)563-3807.

—New Jersey—

New Jersey Software Services: Full range of CP/M, S-100 services.
- System design
  - Business applications
  - Real-time systems
  - Mathematical analysis
- Software creation/customizing
  - 8080 assembly language
  - Z-80 assembly language
  - BASIC
  - FORTRAN
- Product evaluation/selection
  - Hardware Software
- In house training
- Telecommunication service
- Voice I/O applications

Contact C. A. Ryan, 6 Village Circle, Westfield, N.J. 07090 (201) 233-9297.

—Washington—

CHI ENERGY: Custom programs and package modification in Assembler, Basic & C languages; CP/M and real time systems. Contact: Mark A. Carlson, P.O. Box 55145, Seattle, WA 98155. (206)364-5463

Software Directory, continued...

single keystroke. Similar words may also be listed. After proofreading, the Disc-tionary leaves files containing the original unmarked text (BAK file), the marked text, and an alphabetized list of misspellings. As distributed, nearly 50,000 words can be recognized, and expansion allows up to four times that many. Many options including automatic suffix removal are available. All functions are performed by a single menu-driven program for ease of use. Two free "bug-fix" updates are included in the price.
Release: September 1981
Price: $79.00
Included with price: User manual, 8" diskette (manual available separately for $15.00).
Where to purchase it:
Stellarsoft Corporation
841 Blanchette Dr.
East Lansing, MI 48823
(517)332-2459

Program Name: Utilities Software Disk (DMM-1)
Hardware System: CP/M 2.x or MP/M system
Minimum Memory Size: 16K
Language: Object Code
Description: Disk contains the following programs:

XDIR: Displays disk directory file names in alphabetic order and size of each file name. Also shown are the number of bytes on disk, number of file names in use, space used, number of available file names and space. Works on single and double density floppy disks as well as with hard disks.

EXTRACT: Will list a portion of a file between two label names. You do not have to list out whole file.

STRIP: Removes hex code from a PRN file and turns it back into an ASM file.

SORT: Creates symbol table from an assembly done with ASM that can be listed or used with Digital Research debugger SID.

CONVERT: Changes all uncommented lower case characters to upper case. Handy for nice looking listings and for assemblers that will not accept lower case.

STATUS: Provides system information such as memory available, TPA size, top of memory address, I/O assignments and more.
Release: September 1981
Price: $35 plus $1.50 shipping and handling
Included with price: Disk, 8" single density or 5-1/4" North Star
Where to purchase it:
Elliam Associates
2400 Bessemer St.
Woodland Hills, CA 91367
(213)348-4278 ■

CP/M SYSTEMS
COMPATABLE 8080/Z80 SOFTWARE

		$	Source/Object
	STANDARD UTILITIES		
COMMX T.M.	— Menu Driven Communications with Information Services (DEC - IBM - UNIVAC - CBBS - etc.) and Remote COMMX Systems Transfer Any File Type/Size. Nine Link and Eight Local Functions Prompt for Mindless Operation.	250.00	75.00
D	— Disk Directory 4 Column Sort with File Size/Disk File and Space Status.	40.00	20.00
DDB	— Disk Directory Database UPDATE/INQUIRY Catalogs Files Fast.	60.00	25.00
DCOMP	— Disk File Compare with Another Disk File with Display Option.	30.00	20.00
MCOMP	— Memory Range Compare to Memory (ROM or RAM) - Console Logs Errors.	30.00	20.00
MTEST	— Memory Test Any Range with Before/After Write Error Bits + Pass #	30.00	20.00
	ADVANCED UTILITIES		
CDIR	— Comprehensive Sorted Disk Directory/Cross File Block Allocation Check.	30.00	20.00
COPSEQ	— Specify Disk Area and Copy Sequentially to CP/M File.	30.00	20.00
DASM	— 8080 Object Dis-Assembler with Symbol Table/XREF/ASCII MAP.	100.00	40.00
DXRSIZ	— Disk Exerciser Read or Write/Track/Sector/All/Set and Check Skew.	60.00	25.00
GEDIT	— Gang String Substitution Made Globally in One Pass Editor.	50.00	20.00
PREDIT	— Source Program Version Number Maintenance at Pre-Edit Time.	40.00	20.00
PROMER	— Load/Display/Patch/Copy/Verify/Burn ½K+1K+2K+4K Proms.	60.00	30.00
RELOC	— 8080 Object Code Relocator: Put This Into Your Program.	30.00	20.00
X6502	— 6502 Crossassembler MAC Macro Library and Post Processor.	100.00	
	MEMORY MAPPED VIDEO		
CGEN	— EPROM Character Generator Editor for Video Display Boards.	50.00	20.00
DXAM	— Disk Track Sector Examine with Update in HEX or ASCII or EBDIC.	40.00	20.00
VBASIC	— 9K Disk Basic with Super Video Commands and Full Screen Program Editor. Supports Different Video Cards with Identical Program Execution!	CALL	100.00
VGAMES	— For VBASIC: Othello/Blackjack/Breakout/Blockade/Poker Slot and Draw.	100.00	50.00
SOUNDS	— VBASIC Development System for AY-3-8910 Sound Chip Sounds.	75.00	30.00
PMIS	— Program Management Information System (Critical Path Method).	200.00	90.00
DBMS	— VBASIC Data Base Management System/Define/Enter/Report.	150.00	75.00
VIDEO	— Parameter Controlled Multi-User or Scroll Window Video Driver.	50.00	
VDRAW	— Vector Line Draw and Plot Subroutine for Fast Graphics.	30.00	
CHESS	— Graphic Games: IMSAI VIO: VECTOR GRAPHIC Flashwriter 2: SSM VB3.	30.00	20.00
INVADERS	— Zap!/Sound Effects/Joystick or Buttons or Console/Kill or be Killed.	50.00	20.00
STARTREK	— Realtime Action/Sound Effects with Host of Commands and Missions.	40.00	20.00
TARGET	— Moving Aircraft Shooting Gallery with Speed Options/Sound Effects.	40.00	20.00

*CP/M is a Registered Trademark of Digital Research

Disk $7.50 Extra — Cal. Residents Add 6% Sales Tax
Send Your Disk! — S. Den 8" +MICROP+APPLE+NSTAR
Dial 213/348-7909 to Get Free Product Brochure

HAWKEYE GRAFIX T.M.

**23914 MOBILE
CANOGA PARK
CA 91307 USA**

Get 12 issues of Creative Computing for the price of 8.

Some things are still cheaper by the dozen.

When you subscribe to *Creative Computing*, you get 12 issues for just $20. The same 12 issues would cost you $30 at the newsstand.

Why not enjoy *Creative Computing* all year long and save $10 at the same time.

To subscribe, call toll-free from 9 AM to 6 PM 800-631-8112. In New Jersey, call 201-540-0445. Or write to Creative Computing, Morris Plains, NJ 07950. We accept Visa, MasterCard and American Express.

Creative Computing is the leading magazine of small computer applications and software. It has in-depth reviews of new systems, peripherals and software. Also articles for both beginners and experts; columns about popular computers, programming techniques and new products; and complete program listings for your computer.

Alvin Toffler says, "I read *Creative Computing* not only for information about how to make the most of my own equipment but to keep an eye on how the whole field is emerging."

Why not join over 90,000 subscribers and save money at the same time? If you're clever enough to order a dozen.

Creative Computing
the #1 magazine of computer applications and software
March
vol 7
Emphasis on Education
Simulation of Competitive Business
MECC: An In-depth Report
Seymour Papert: Mindstorms
How to Select a Computer
How to Solve It
CAI Is Not Always Drill
Reading Level
Evaluation of
25 Fantastic New Products from the Consumer Electronics Show
Creative Computing software

NEW PRODUCTS

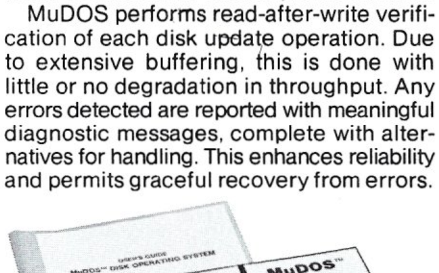
Multi-User Software Offers Extra Performance And Features For CP/M, MP/M, CP/Net-type Systems

MuSYS Corporation, maker of multi-user microcomputer systems, is introducing MuDOS, a CP/M compatible operating system offering higher throughput, increased reliability and extra professional features for both single and multiple-user environments. The system may be customized to any Z80-based hardware configuration and used in place of CP/M, MP/M and CP/Net. Program loading under MuDOS is six times faster than CP/M. File processing functions average three to five times faster. MuDOS uses the extra registers and instructions available on the Z80 to speed processing of operating system calls.

Many standard features on MuDOS are either optional, extra cost or on available on CP/M. The most significant of these extra features are a buffer manager, a totally re-entrant file manager and an optional multiple print queuing capability. A sophisticated buffer manager reduces greatly the number of physical disk accesses required. The amount of buffering is a user parameter, and may be changed dynamically by the application program. A re-entrant file manager allows simultaneous accesses on different controllers, if the controller hardware supports DMA transfers. Also, higher priority tasks may interrupt long file operation. Optional automatic, concurrent print spooling, allows use of multiple print queues, forms types, fonts, and even hand-fed single sheet printing. This option satisfies a wide variety of data and word processing requirements.

MuDOS supports disk files up to 67 megabytes and drives up to 2000 megabytes and is usable with today's most advanced disk systems. Warm start and disk log-on displays are unnecessary. Allocation information is recorded on the disk. Disks may be changed any time updating is not in process. This eliminates disk system resets or warm starts which are particularly bothersome in other multi-user implementations.

MuDOS performs read-after-write verification of each disk update operation. Due to extensive buffering, this is done with little or no degradation in throughput. Any errors detected are reported with meaningful diagnostic messages, complete with alternatives for handling. This enhances reliability and permits graceful recovery from errors.



Modular architecture permits easy adaptation to different user environments. Module commonality ensures compatibility between different MuDOS versions. A relocating, linking, loader program determines system memory size, selects the configuration specified in a symbolic parameter file, and relocates, links, and loads the modules. Up to 30 modules are available in the resident portion of MuDOS, depending on system configurations elected. Each hardware dependent element is held within a separate relocatable module. Modules may be changed or replaced without massive reasasemblies or system generations.

Price: MuDOS (diskette and User's Guide) $300-$750 depending on configuration. For more information, contact Mr. Bill Schultz, MuSYS Corporation, 1451 E. Irvine Blvd., Suite 11, Tustin, CA 92680. (714)730-5692. TWX: 910-595-1967. Cable: MUSYSTSTN.

CompuPro Announces 64K Static RAM

CompuPro division of Godbout Electronics announces the RAM 17—a 64K fully static memory board for S-100/IEEE 696 computers. It is the highest density static RAM ever offered for the S-100/IEEE 696 bus on a standard 5 inch high board, and offers many advantages over its dynamic counterparts. Among them are faster access time guaranteed to run with 6 Mhz Z-80's and 10 Mhz 8086/88's), lower power dissipation (less than 2 watts), freedom from alpha particle soft-bit errors, and problem-free DMA operation.

It meets all the IEEE 696 specifications including 24-bit addressing, allowing up to 16 Mbytes of system memory, may be addressed on any 64K page boundary, and can be disabled in 16K blocks. The upper 8K block can also have 2K windows disabled to allow for memory-mapped peripherals. It uses the new 2K x 8-bit static RAM chips that are pin-compatible with 2716 type EPROM's, allowing up to 32K or EPROM to intermix with the RAM.

S-100
MICROSYSTEMS
NOV/DEC 1980
VOL. 1/NO.6
IMPLEMENTING THE CP/M IOBYTE FUNCTION
See Pages 22-26
the CP/M and S-100 user's journal
MICROSYSTEMS
VOL.2/NO.3
MAY/JUNE 1981
THE COMPUTER COMMUNICATIONS EXPLOSION
See Pages 20-28

Microsystems — the CP/M* and S-100 User's Journal

CP/M is the software bus!+ S-100 is the hardware bus for sophisticated microcomputer users!

If you are a CP/M user, on any system—S-100, Apple, TRS-80, Heath, Ohio Scientific, Onyx, Durango, Intel MDS, Mostek MDX, etc—after all CP/M is the Disk Operating System that has been implemented on more computer systems than any other DOS—then *Microsystems* magazine is the "only" magazine published specifically for you!

Or, if you use an S-100/IEEE-696 based computer—and the most sophisticated microcomputer systems available use the S-100/IEEE-696 hardware bus—then *Microsystems* magazine is the "only" magazine published specifically for you!

We started publishing *Microsystems* almost two years ago to fill the void in the microcomputer field. There were magazines catering exclusively to the TRS-80, Apple, Pet, Heath, etc. system users. There were also broad based publications that cover the entire field but no one system in depth. But no magazine existed for CP/M users—nor did one exist for S-100 users.

The why and what of a software bus

First of all what is a "bus?" And why do we call CP/M "the software bus?"

A "bus" is a technique used to interface many different modules. Examples are the "S-100/IEEE-696 Bus" and the "IEEE-488 Bus." These are hardware buses that permit a user to plug a bus-compatible device into the bus without having to make any other hardware modifications and expect the device to operate with little or no modification.

CP/M is a Disk Operating System (DOS). It was first introduced in 1974 and is now the oldest and most mature DOS for microcomputer systems. CP/M has now been implemented on over 250 different computer systems. It has been implemented on hard disk systems as well as floppy disk systems. It is supported by two user groups (CP/M-UG and SIG/M-UG) that have released over 80 volumes containing over 2,000 public domain programs that can be loaded and run on systems using the CP/M DOS. Add to this another 1,500 commercially available CP/M software packages and you have the largest applications software base in existence.

CP/M is the only DOS for micros that has stood the test of time (seven years) with the highest level of compatibility from version to version. And over the years this compatibility has been maintained as new features have been added.

This is why we say "CP/M is the software bus" and why *Microsystems* magazine is vital to providing CP/M users with technical information on using CP/M, interfacing to CP/M, new CP/M compatible products and for CP/M users to exchange ideas.

Why support the S-100 bus?

S-100 is currently the most widely used microcomputer hardware bus. It offers advantages not available with any other microcomputer system. Here are a few of the advantages:

S-100 is processor independent. There are already thirty different S-100 CPU cards that can be plugged into an S-100 bus computer. Nine 8-bit microprocessors are available: 6502, 6800, 6802, 6809, 2650, F8, 8080, 8085 and Z80. Eight 16-bit microprocessors are available: 8086, 8088, 9900, Z8000, 68000, Pascal Microengine, Alpha Micro (similar to LSI-11) and even the AMD2901 bit slice processor. Take your pick from the incredible offerings.

S-100 has the greatest microcomputer power. What other microcomputer system has direct addressing of up to 16 megabytes of memory, up to 65,536 I/O ports, up to 10 vectored interrupts, up to 16 masters on the bus (with priority) and up to 10 Mhz data transfer rate? You will have to go a long way to use up that computing power.

S-100 is standardized. The S-100 bus has been standardized by the IEEE (Institute of Electrical and Electronic Engineers) assuring the highest degree of compatibility among plug-in boards from different manufacturers. And, *Microsystems* has published the complete IEEE S-100/696 standard (all 26 pages).

S-100 has the greatest hardware support. There are now over sixty different manufacturers of about 400 different plug-in S-100 boards. Far greater than any other microcomputer system.

With all these advantages is it any wonder that S-100 systems are so popular with microcomputer users who want to do more than just play games?

For the serious computer user.

Each issue of *Microsystems* brings you the latest in the CP/M and S-100 world. Articles on applications, tutorials, software development, product reviews, and lots more, to keep you on top of the ever changing microcomputer scene.

And if you are an S-100 system user using other operating systems (e.g. North Star) *Microsystems* also supports you.

Get your copy today

Order your susbscription to *Microsystems*. Send $10 for one year (6 issues), $18 for two years (12 issues) or $24 for three years (18 issues). If you prefer, call our toll-free number, **800-631-8112** (in NJ 201-540-0445) to put your subscription on your MasterCard, Visa or American Express card. Canadian/Mexican and other foreign surface subscriptions are $15 and $25, respectively, per year and must be prepaid. We guarantee that you will be completely satisfied or we will refund your subscription.

Join thousands of *Microsystems* subscribers like Jim Johnstone of Los Altos CA, who said "*Microsystems* has lived up to your promises and my expectations. Congratulations."

the CP/M* and S-100 user's journal
MICROSYSTEMS
a Creative Computing publication

39 East Hanover Avenue
Morris Plains, NJ 07950, USA
Toll-free 800-631-8112
(In NJ, 201-540-0445)

New subscribers are entitled to a free CP/M Reference Card. Be sure to ask for yours when you order.

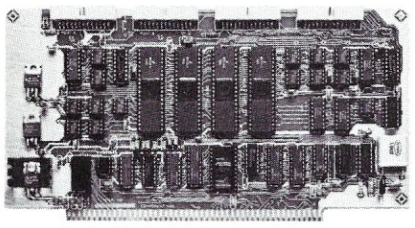
* registered trademark Digital Research + registered trademark Lifeboat Associates

New Products, continued...

Available in Unkit, Assembled and Tested or Certified System Component high reliability program. Single quantity end-user prices are $1095, $1395 and $1595 respectively. Also available in a 48K configuration. CompuPro division of Godbout Electronics, P.O. Box 2355, Oakland Airport, CA 94614.

S-100 4-Port Serial I/O Interface

The CCS 2710 4-Port Serial I/O Interface permits independent programmable port control using four 8250 asynchronous communications elements with extensive control over I/O format and operation. Each port has full handshaking. The system interface



meets IEEE proposed S-100 standards. Word length, parity, modem, number of stop bits, and interrupt conditions are all software-determined. The baud rate may be programmed for any between 2K and 56K baud.

Driver routines and/or other programs may be stored on-board in a user-supplied 2K EPROM. The base address of the ROM is jumper-selectable at any 2K boundary. If the phantom output is jumper-enabled, the ROM routines memory overlay sharing the ROM's addresses.

For more information, contact: California Computer Systems, 250 Caribbean Drive, Sunnyvale, CA 94086, (408)734-5811.

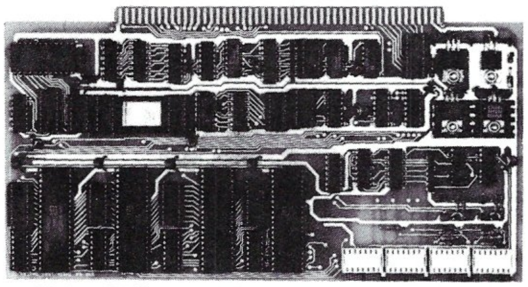
Cromemco-Compatible 64K RAM Board

SUPERAM 4C is a 64K dynamic RAM board functionally equivalent to Cromemco's 64KZ RAM card. No user training or jumpers are required to achieve compatibility because the operation of all of the diagnostic LED's and user switches is identical. The board is organized in two blocks of 32K bytes. Each 32K block can be placed in any of eight different memory banks. Bank selection is available on 32K byte boundaries. Address and bank assignment of each 32K block is selectable via simple switch settings. Up to eight boards can be used with bank select to expand memory from 64K to 512K. Memory refresh is totally transparent to the processor. Memory access time is 250 nsec. Operating speed is 4MHz with no wait states. The price of the board is $995. Piiceon Inc., 2350 Bering Drive, San Jose, CA 95131; (408)946-8030.

MULTI USER?

We have Multi Ports!

4 Serial RS-232 I/O Ports for the IEEE S-100 Bus



THE SIO-4A BOARD HAS:

—Current loops on 2 ports
—4 Asynchronus UARTs
—Status handshake for fast printers
—Baud rates from 4.7 Baud to 38.4K
—Crystal time base
—One year warranty

$249.50 - Assembled & Tested

THETA LABS, INC

P.O. Box 20337/Dallas, Texas 75220
(214) 351-6093 *contact Diane*
Dealer Inquires Welcome

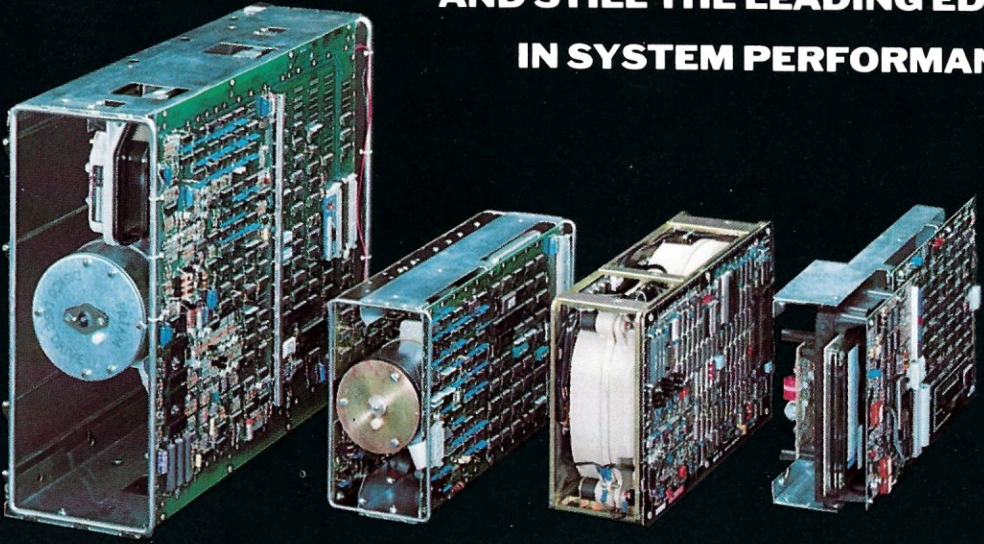
Advertiser Index

advertiser	page
ABM Products	10
Ackerman Digital Systems	31
Adaptive Data & Energy	cover 3
Alpha Data Services	65
Bower-Stewart & Assoc	58
California Digital Engineering	29
Casheab	56
Computer Design Labs	23
Computer Services Corp.	13
Computer Innovations	41
Computing Interface	6
Cornerstone Software	74
Creative Computing	77
Digital Constructs	52
Digital Marketing	9
Dual Systems Control Corp.	7
Ecosoft	67
Electronic Control Technology	19
Electronic Specialists	16
Executive Data Systems	73
Godbout Electronics	cover 4
Hawkeye Grafix	76
Hayes Microcomputer Products	63
Information Technologies	63
Infosoft Systems	45,57
Integrand	74
Interactive Microware	41
JES Graphics	63
Knowlogy	71
Laboratory Microsystems	35
Lomas Data Products	25
Magnolia Microsystems	43
Marketing Essentials	21
McClintock Corp	71
MicroDaSys	1
Micro-Logic, Inc.	11
Microsystems	79
MicroTech Exports	45
Midwest Micro Warehouse	14
Morrow Designs	cover 2
Mullen Computer Products	17
PCE Systems	15
Phila. Area Computer Show	75
Potomac Micro-Magic	10
RR Software	5
S-100, Inc.	67
Small Business Computers	39
Snow Micro Systems	14
Software Review	73
Southern Computer Systems	6
SSM Microcomputer Products	2
Static Memory Systems	55
Stellarsoft	45
Systems Plus	67
SZ Software	14,45
TecMar	49
Technical Software Systems	18,38
The Code Works	69
The Rosetta Stone	73
The Software Connection	61
Theta Labs	80
Timin Engineering	69
Tiny C	69
Western Wares	69
WW Component Supply	33

THE HARD EDGE

AND STILL THE LEADING EDGE . . .
IN SYSTEM PERFORMANCE

ADES

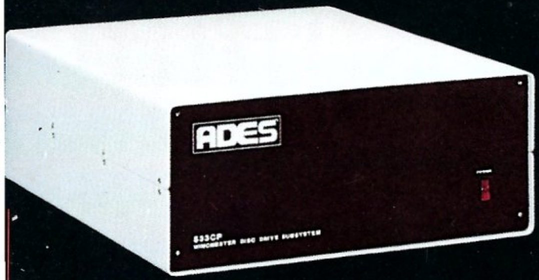


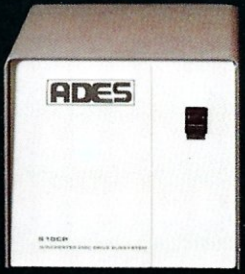
Hard Disk and Streaming Tape

reliability and versatility that can't be beat

OEM CONTROLLERS

- Multibus
- General Purpose
- S100





S33

• 31 or 62 MByte formatted

S10

• 10.5 or 32 MByte formatted

• Lowest cost/MByte from any manufacturer
• CP/M* compatible "drop in" BIOS
• Single S100 card controller
• Fully expandable up to four drives, 8" or 14"
• Reliable high performance Priam Winchester Disks
• Immediate delivery

SYSTEM 8

• 8" Winchester 10 or 32 MByte formatted
• Integrated streaming cartridge backup
• Streaming backup at 5 MB/min
• Selective file backup under CP/M
• Versatile parallel 1/0 or DMA interface

ADES

ADAPTIVE DATA & ENERGY SYSTEMS

2627 Pomona Boulevard • Pomona, CA 91768

Phone: (714) 594-5858

CP/M is a trademark of Digital Research Corp.

Why do so many computer companies buy CompuPro components?

Simple: They need professional-level S-100 computers that can grow with their needs, whether for testing, research, or development.

Our customers range from Osborne Computers, one of the newest companies around, to IBM, the world-wide business computing veterans...from the author who needs a word processor that's better than the rest, to software companies who depend on **CompuPro** components to develop the software that's their bread and butter.

We didn't get these kinds of customers by following the crowd. Instead, we earned our reputation through technological leadership that started seven years ago and continues today. While others are still talking about 16 bit operation, we manufactured and delivered a CPU board using Intel's 8085 and 8088 (capable of handling both 8 **and** 16 operation) back in June of 1980. And when it comes to high speed S-100 operation, no one does it faster; not only do we make the industry's fastest CPU boards, but our memories, disk controllers, motherboards, interfacers, and accessory boards are all designed from the ground up to work efficiently (and transparently) in ultra-high speed systems.

Of course, tomorrow's technology will bring even higher standards of performance. However, since **CompuPro's** fully modular systems are easily expandable, you can increase computing power - at any time - simply by adding to the system you already have. Upgrade from single to multi-user operation, or from 8 to 16 bits; the choice is yours. This modularity protects your initial investment, and allows your computer to grow as your requirements grow.

Add an enviable reputation for reliability (and one of the best warranties in the business), and it's easy to see why professional systems integrators choose **CompuPro** for demanding commercial, industrial, and scientific applications. When you need the same kind of computing power and ability to upgrade as those who depend on computers for a living, **CompuPro** systems and components are available at finer computer stores world-wide.

CompuPro™ division of GODBOUT ELECTRONICS

OAKLAND AIRPORT, CA 94614 (415) 562-0636